ISSN 0542 9951

8
1996

NAJSTARSZE PISMO INFORMATYCZNE W POLSCE – UKAZUJE SIĘ OD 1965 ROKU

informatyka

miesięcznik profesjonalistów

POLITECHNIKA ŚLĄSKA Biblioteka Główna

P. 1877/96

Dojrzewanie do stopni zawodowych

Lesław Wawrzonek

O stopniach zawodowych informatyków dyskutuje się już w Polsce od 1981 roku. O powodzeniu tego projektu zadecyduje siła środowisk zainteresowanych jego sukcesem lub porażką. Przyjrzyjmy się układowi sił, od których może zależeć efekt końcowy.

Zawsze, gdy ktoś chce wprowadzić u nas trochę ładu w dowolnej dziedzinie, znajdą się przeciwnicy, którzy usiłują zdyskwalifikować pomysłodawcę używając demagogicznych argumentów i epitetów. To dla nich Witkacy wymyślił termin – spłyciarze. W przypadku stopni zawodowych informatyków w jakiejkolwiek formie zwalczają je oczywiście ci, którzy na ich wprowadzeniu mogliby dużo stracić. Należą do nich wyedukowani na grach i pakietach biurowych „specjaliści", którzy nie pokonaliby żadnej rzetelnej bariery weryfikującej ich umiejętności w dowolnej dziedzinie informatyki. Chcieliby, co jest zrozumiałe, jeszcze długo podawać się za konsultantów czy specjalistów.

Po drugiej stronie są specjaliści, którzy mogliby na wprowadzeniu systemu stopni zawodowych wiele zyskać. W ciągu wielu lat pracy zdobyli rzetelną wiedzę i duże doświadczenie i formalne potwierdzenie tego faktu nie stanowiłoby dla nich problemu. Spodziewają się oni, obserwując sytuację w dziedzinach o ustalonej strukturze specjalizacji, że posiadanie zawodowego certyfikatu mogłoby być dla nich bardzo korzystne. Zwłaszcza, jeżeli będzie on w miał w przyszłości zasięg ponadnarodowy.

Zwolennikami stopni zawodowych są też absolwenci uczelni, którzy ciągle podnoszą swoje kwalifikacje. Absolwent uczelni po pięciu czy dziesięciu latach jest wart na rynku pracy tyle, ile dokonał w okresie od skończenia studiów. Czułby się pewnie bezpieczniej, gdyby mógł potwierdzić swoje umiejętności zdobytym w tym czasie stopniem zawodowym.

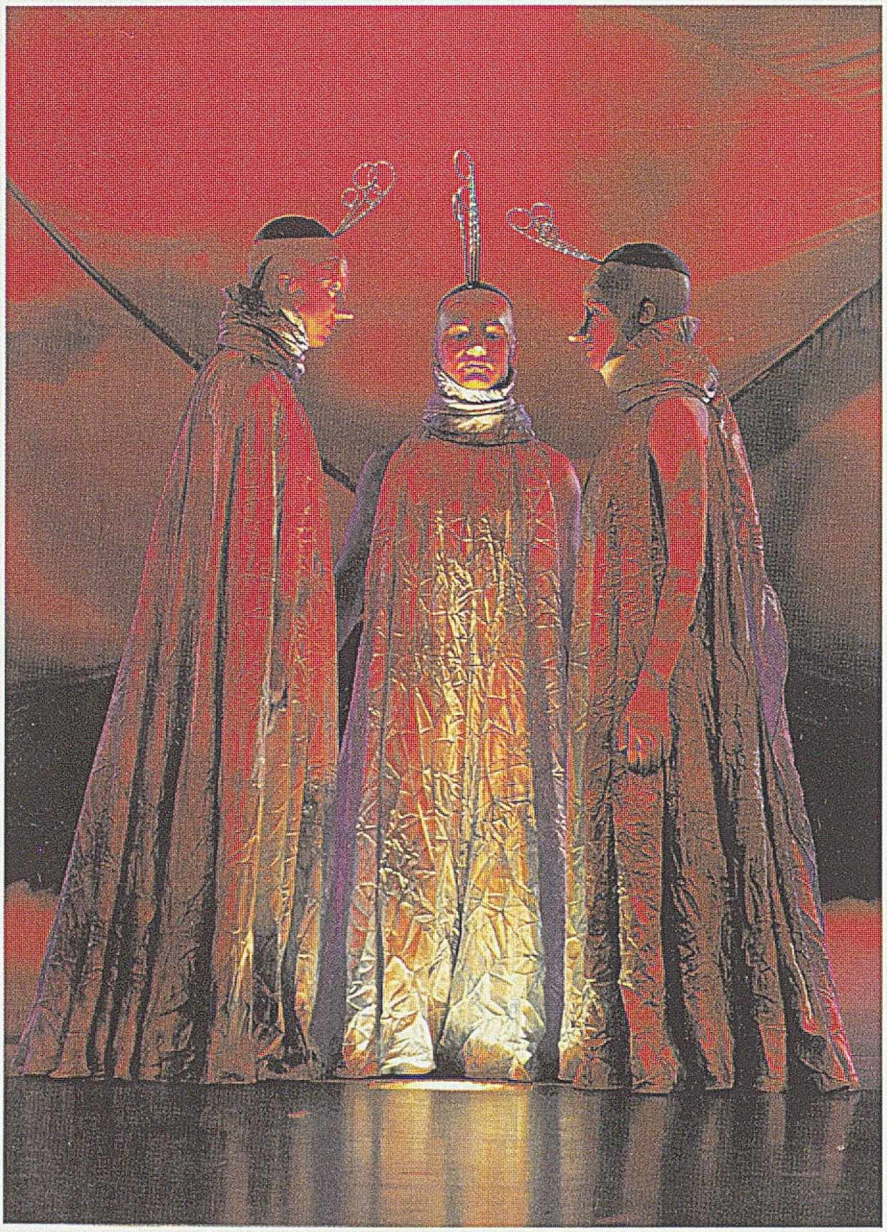
Zarządy firm komputerowych są niechętne lub w najlepszym razie obojętne wobec perspektywy powstania systemu stopni zawodowych informatyków. Nie będą więc skłonne inwestować w taki projekt. Podczas przyjmowania do pracy etatowej lub na terminowy kontrakt, udokumentowana formalnie wiedza ułatwia wybór właściwego kandydata. Z drugiej strony etatowy pracownik z uznawanym wszędzie certyfikatem zawodowym jest mniej uzależniony od pracodawcy.

Odbiorców i użytkowników systemów informatycznych stopnie zawodowe informatyków dotyczą tylko pośrednio. Woleliby oni jednak aby ewentualny kontrahent, partner lub wykładowca legitymowali się formalnie udokumentowaną wiedzą. Przeznaczają bowiem często znaczne środki na komputeryzację i obawiają się, żeby nie zostały one zmarnowane. Jak wykazuje doświadczenie nie zawsze są to płonne obawy. Pilnie też potrzebują „potwierdzonych" specjalistów-informatyków sądy, policja, wojsko, komisje przetargowe, administracja państwowa, komisje arbitrażowe. Gdy w środowisku informatyków i Polskim Towarzystwie Informatycznym trwa dyskusja, niezależnie zaczynają powstawać inne formy certyfikatów zawodowych w odpowiedzi na takie zapotrzebowanie. Na przykład Polska Izba Informatyki i Telekomunikacji pracuje obecnie nad trybem nadawania specjalizacji, stopni i uprawnień swoim rzeczoznawcom. Podobne plany ma Stowarzyszenie Użytkowników Systemów Inżynierskich. Okazuje się, że najlepiej wprowadzać ceryfikaty zawodowe tam, gdzie powstaje na nie wyraźne zapotrzebowanie.

Mimowolnymi wrogami stopni zawodowych okazali się sami organizatorzy całej akcji. Ich błąd polega na zamiarze natychmiastowego wdrożenia gotowego systemu, który w Wielkiej Brytanii powstawał przez wiele lat, tworząc w tym czasie strukturę administracyjną, kadrę ją obsługującą i tradycję. Skoro istnieje znaczna różnica poziomu zorganizowania i wielkości środków między British Computer Society a Polskim Towarzystwem Informatycznym, należało ten fakt uwzględnić, przejmując działający tam system stopni zawodowych.

Większość informatyków jeszcze sobie nie uświadomiła, że wprowadzenie stopni specjalizacji zawodowych jest dla każdej grupy zawodowej bardzo skutecznym sposobem zabezpieczenia indywidualnych interesów jej członków. Sposób, w jaki się do tego zabierają, świadczy o tym, że społeczność polskich informatyków jest jeszcze bardzo słabo zorganizowana. Jeżeli dodać, że British Computer Society powstało trzy lata później niż Polskie Towarzystwo Informatyczne, to robi się naprawdę smutno.

Zbliżenie, złączenie ... Integracja !



Coraz więcej do zrobienia, coraz mniej czasu ... Dajemy Waszym firmom coś, co sprawi że nagła awaria nie uwięzi Was znowu w firmie po godzinach. ***ALPHA*** serwery i stacje robocze wyposażone w WINDOWS NT, DIGITAL UNIX bądź w OPEN VMS. Możecie wybrać orginalny sprzęt ***ALPHA DIGITAL*** lub znacznie tańszy ***ALPHA HECTOR***. Chcecie mieć jeszcze więcej czasu ? Tworzymy integracje systemów, dajemy techniczny support. Mało ? Kompleksowo instalujemy sieci, również ***INTERNET*** ! A teraz coś o bezpieczeństwie w waszej firmie. Jest zapewne wielu ludzi, którzy pragną włamać się do waszego systemu przez INTERNET. Teraz możecie się przed tym zabezpieczyć. Proponujemy najskuteczniejsze zabezpieczenie : ***FIRE WALL*** firmy DIGITAL. To naprawdę działa ! Przekonaj się sam. Coraz więcej do zrobienia... coraz więcej czasu !

hector S.A.

ul. Gwiaździsta 19 01-651 Warszawa
tel. + 48 (22) 330 285 333 967 333 992
kom. 3912 3065 fax 333 964 bbs 337 258
http://www.atm.com.pl/~hector/hector.htm

publikacje

POLITECHNIKA ŚLĄSKA Biblioteka Główna Biblioteka
P. 1877/96

OB! OK! - obiektowy system zarządzania bazą danych

Maciej Piasecki
Centrum Informatyczne Politechniki Wrocławskiej

OB!OK! (skrót od nazwy „Object Base! OK!") to obiektowy system zarządzania bazą danych (w skrócie OSZBD) skonstruowany w oparciu o model obiektowy wzorowany na modelu stosowanym w imperatywnych, obiektowych językach programowania (np. *C++*, *Pascal*). System *OB!OK!* został zrealizowany we wrocławskiej firmie *Bogart sp.zo.o.* Zarys opisu formalnego przyjętego modelu oraz analizę jego zalet i niedomagań można znaleźć w artykule: „Koncepcja modelu dla obiektowej bazy danych" [3]. Obecnie zostaną omówione rozwiązania techniczne, realizujące w *OB!OK!* zaprezentowany tam model.

Geneza powstania systemu

Dwa główne powody zadecydowały o podjęciu decyzji zrealizowania komercyjnej wersji własnego OSZBD.

- Pierwszy był względem praktycznym. Firma Bogart podjęła się realizacji dużego multimedialnego systemu informacyjnego *SMOK* (System Informacyjny o Mostach Kolejowych) dla *PKP*. Złożoność projektu oraz zastosowanie multimediów jako integralnej części przyszłego systemu informacyjnego wskazywało na nieadekwatność wykorzystania tu jednego z istniejących relacyjnych systemów zarządzania bazami danych (RSZBD). Z drugiej strony istniejące obecnie OSZBD są zbyt mało wydajne i stawiają zbyt duże wymagania sprzętowe [3].

- Drugim powodem była nadarzająca się możliwość realizacji pasji badawczej autora.

Na projekt systemu *OB!OK!* narzucono dwa ostre wymagania: posiadanie podstawowych cech OSZBD z jednoczesną efektywnością pracy na komputerach klasy PC oraz przeznaczenie do zastosowań komercyjnych. Całość przedsięwzięcia pochłonęła ok. 8 „osobomiesięcy".

Sformułowane powyżej wymagania uwzględniono na każdym etapie prac nad *OB!OK!*, począwszy od sformułowania koncepcji obiektowego modelu systemu zarządzania bazą danych (w skrócie SZBD).

Realizacja modelu OSZBD w *OB!OK!*

Każdy SZBD jest zbiorem środków realizacyjnych do budowy systemu informacyjnego. Zwykle SZBD jest również niezbędnym elementem środowiska działania takiego systemu. Prawie zawsze kompletny system informacyjny składa się z trzech komponentów:

1. Struktura fizyczna danych
2. SZBD
3. Aplikacja

Pierwszy komponent, aplikacja zaprojektowana dla konkretnych zadań operuje na danych wyrażonych przez pojęcia zdefiniowane w modelu logicznym i usługi udostępniane przez SZBD. Komponent drugi to SZBD, który ustanawia model logiczny danych będący podstawą jego działania oraz udostępnia operacje na danych. Zaś trzeci – to struktura fizyczna plików przechowujących dane na dysku. Komponenty wyznaczają zatem schematyczną strukturę systemu informacyjnego z trzema poziomami postrzegania danych. Przy budowie systemów OSZBD problemem jest niespójność modelu logicznego SZBD z abstrakcją danych wewnątrz aplikacji.
W *OB!OK!* modelem logicznym SZBD jest **model obiektowy** [3], w którym pojęcie **obiektu** tożsame jest z programistycznym pojęciem obiektu. Dzięki temu nie ma wspomnianej wcześniej niespójności. Zarówno aplikacja, jak i OSZBD pracują w oparciu o ten sam **model logiczny**. Zachodzi tylko jeden etap translacji danych ze struktury fizycznej do struktury abstrakcyjnej. Kształt tej struktury wynika z przyjętego **modelu obiektowego**. Mimo zgodności definicyjnej obiektów składających się na dane zarządzane przez *OB!OK!* oraz obiektów programistycznych (*C++*) tworzących aplikację, oraz tworzących OSZBD, to za pomocą kryterium funkcjonalności można dostrzec dwie ich grupy. Pierwszą grupę nazwiemy **obiektami danych**, a dla drugiej będziemy używać terminu **obiekt**.
Zdefiniowany w [3] **model obiektowy** warto uzupełnić o pojęcie:
świat danych – zbiór **klas** pozostających ze sobą w relacjach semantycznych, stanowiących model pewnego rozpatrywanego wycinka rzeczywistości.

Pojęcie to na poziomie modelu logicznego jest bliskie pojęciu *widoku* (znanemu z RSZBD), natomiast na poziomie struktury fizycznej (poziom III) jest odrębnym zestawem plików danych.

Zarządzanie danymi

Najmniejszą częścią informacji przechowywanej w bazie danych (na poziomie modelu logicznego) jest **obiekt danych** o strukturze zgodnej z **modelem obiektowym** *OB!OK!*. **Obiekty danych** są pogrupowane **w światy danych,** a w ich obrębie w **klasy**. Każdy **obiekt danych** w momencie utworzenia (za pomocą odpowiedniej usługi systemu) otrzymuje swój **identyfikator**. Identyfikator jest unikalny w danym **świecie danych**. Unikalność **identyfikatora** jest zagwarantowana do końca istnienia **obiektu danych**. Dzięki **identyfikatorowi** jest możliwe operowanie **obiektem danych** za pomocą usług systemu (uzyskiwanie dostępu, zmiana stanu albo zniszczenie **obiektu danych**).

Obiekty danych dowolnej z **klas** możemy uporządkować przez relację, otrzymując w ten sposób *uporządkowaną kolekcję*. Abstrakcja kolekcji i *relacji*, będącej jej podstawą, jest realizowana fizycznie przez mechanizm indeksowania. Daje to możliwość szybkiego przeglądania **obiektów danych** danej **klasy** według przyjętego kryterium. Całość tego rozwiązania, tzn. pojęcie kolekcji, relacji oraz zbioru indeksowego będziemy w skrócie określali mianem **indeksu**.

Przestrzeń możliwych typów obiektowych

Każdy **typ obiektowy** definiowany w *OB!OK!* musi spełniać schemat określony przez przyjęty **model obiektowy** [3]. **Model obiektowy** *OB!OK!* zawiera również zestaw *typów prostych*, które mogą zostać użyte w definicjach **atrybutów obiektów danych**. W tabeli 1 zestawiono typy proste oraz kilka dostępnych predefiniowanych typów obiektowych o charakterze narzędziowym. Łatwo zauważyć podobieństwo osiągnięte w sposób zamierzony do zbioru typów prostych języka *C++*.

Tabela 1. Typy proste oraz typy predefiniowane danych w systemie *OB!OK!*

nazwa typu	opis	nazwa typu	opis
CHAR	typ znakowy (char)	TIME	predefiniowana klasa C++ **CDTime** reprezentująca godzinę (bez daty)
STRING	ciąg znaków (char*)	DATE	predefiniowana klasa C++ **CDDate** reprezentująca datę (bez czasu)
DATE	liczba całkowita (int) (znaczenie takie jak int)	FULLDATE	predefiniowana klasa C++ **CDFullDate** reprezentująca datę i czas
INTEGER	liczba całkowita (int)	GRAPHICS	predefiniowana klasa C++ **CDGraphics** reprezentująca dowolny fragment grafiki (np. rysunku)
LONG	liczba całkowita o podwójnej precyzji (long int)	MULTIMEDIA	predefiniowana klasa C++ **CDMultimedia** reprezentująca dowolny fragment multimediów (np. filmu z dźwiękiem)
DOUBLE	liczba zmiennopozycyjna o podwójnej precyzji (double)	<typ obiektowy>	dowolna klasa danych zdefiniowana w aplikacji
VOID*	wskaźnik na obszar ciąg pełen (nadający się do dowolny blok danych		

Zgodnie z modelem *OB!OK!* typem atrybutu może stać się dowolny typ strukturalny, wywiedziony z predefiniowanych typów prostych (np. tablica, lista, struktura). Model ten wprowadza również możliwość nadawania atrybutowi dowolnego ze zdefiniowanego wcześniej **typów obiektowych**, w tym typów strukturalnych wywiedzionych od nich (np. tablica obiektów, lista albo tablica struktur postaci: obiekt „wartość **INTEGER**"). Otrzymujemy w ten sposób realizację pojęcia **agregacji**, tzn. **składania się** jednego **obiektu danych** (obiekt *ojciec*) z innych **obiektów danych** (obiekty *dzieci*). Relację tę będziemy określać mianem relacji *obiekt – podobiekt*.

Istnienie obiektów dzieci jest ściśle powiązane z istnieniem obiektu ojca. **Destrukcja** dzieci następuje w momencie **destrukcji** ojca.

Do stosowanych w *OB!OK!* typów dodaje się jeden nieznany (dokładniej nieobecny w postaci jawnej w *C++*) element. Jest nim wartość pusta *NULL*. Pojęcie to jest zaczerpnięte wprost z rozszerzonego modelu relacyjnego SZBD [1,2]. Wartość NULL wprowadzamy do zbioru wartości każdego ze stosowanych typów. Semantycznie reprezentuje ona brak wartości.
Powstaje możliwość stosowania *tablicy rzadkiej* (bez niektórych elementów), jako typu **atrybutu**. Nadając **atrybutowi** wartości NULL osiągamy efekt analogiczny do nienadania wartości w ogóle. Zastosowanie drugiej z tych możliwości ma bardzo istotne konsekwencje dla realizacji **indeksu**.

Hierarchia dziedziczenia klas

Klasy definiowane w systemie *OB!OK!* mogą utworzyć *hierarchię dziedziczenia*, tzn. graf acykliczny, w którym każdy następnik ma dokładnie jednego poprzednika (brak dziedziczenia polimorficznego). W myśl definicji **klasa dziedzicząca** przejmuje po innej zarówno definicje atrybutów, jak również definicje metod. Zatem otrzymuje powiązane z nimi **selektory**. Każdy **selektor** atrybutu czy metody jest unikalny w całej hierarchii dziedziczenia.

Każda klasa jest zbiorem obiektów, a każda jej **podklasa** – podzbiorem w nim zawartym. Tworząc kolekcję z elementów **nadklasy** (poprzez definicję odpowiedniej *relacji*) otrzymujemy taką kolekcję, w której skład wchodzą elementy dowolnej podklasy znajdującej się poniżej w hierarchii dziedziczenia. Każda relacja, ponieważ jest metodą, jest dziedziczona przez podklasy. Jej selektor jest widoczny w całym drzewie dziedziczenia poniżej danej klasy. Utworzony w ten sposób **indeks** umożliwia zarządzanie obiektami należącymi do różnych klas.

Nie wszystkie obiekty z klas objętych indeksem muszą być w nim reprezentowane. Można posłużyć się tu pojęciem wartości pustej NULL. Rozważmy najprostszą sytuację: relacja zależy tylko od wartości pewnego wybranego atrybutu (nie jest istotny jego typ, każdy typ zawiera wartość NULL). Wszystkie obiekty, w których rozpatrywany atrybut przyjmuje wartość NULL, nie znajdą się w indeksie (wartość NULL oznacza brak wartości). W myśl formalnej interpretacji, obiekty te znajdą się na końcu (lub początku, co nie ma tu znaczenia) utworzonej kolekcji i mogą zostać w łatwy sposób pominięte podczas przeszukiwania.

Znacznie bardziej skomplikowany jest przypadek indeksu zależnego od więcej niż jednego atrybutu. Mamy wtedy do czynienia z *indeksem złożonym*. Z braku miejsca nie przeanalizujemy dokładnie tego przypadku i ograni-

czymy się do kilku związanych z nim spostrzeżeń. Indeksy złożone w *OB!OK!* mogą być komponowane jedynie z atrybutów obiektu ojca. Nie jest zaimplementowane pojęcie *indeksu ze ścieżką dostępu* [1], które by pozwalało sięgać do atrybutów dzieci. Jednakże ograniczenie to jest tylko pozorne. Zgodnie z ideologią programowania obiektowego, w *OB!OK!* dostęp do atrybutów jest możliwy jedynie poprzez metody obiektu. W *OB!OK!* jest pewna grupa predefiniowanych metod umożliwiających odczyt i zmianę wartości atrybutów. Relacja konstruowana w *OB!OK!* jest zawsze budowana jako złożenie kilku metod. W ten sposób do relacji, stanowiących podstawę konstrukcyjną indeksu, możemy wprowadzić metody operujące na atrybutach obiektów dzieci.

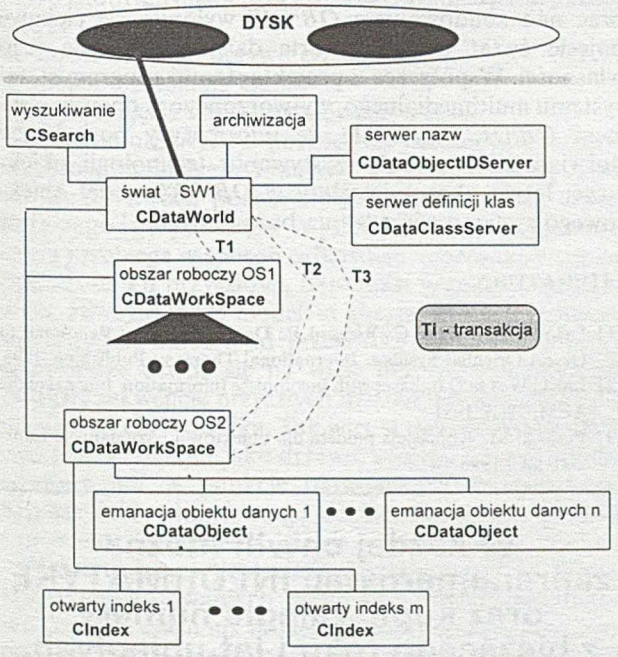
Wybrane aspekty implementacji

Zdefiniowany na potrzeby *OB!OK!* model obiektowy nie ma charakteru „systemocentrycznego" [3]. Mamy tu do czynienia z próbą wprowadzenia jednolitej przestrzeni pojęciowej zarówno dla OSZBD, jak i aplikacji. Dlatego najbardziej naturalnym kierunkiem implementacji jest pogłębianie tej zgodności przez konstrukcję OSZBD w profesjonalnym języku programowania obiektowego, jakim jest *C++*.

OB!OK! w trakcie swojego działania staje się częścią aplikacji. Granica pomiędzy dwoma pierwszymi komponentami staje się niezauważalna. Osiągnięto to poprzez zamknięcie *OB!OK!* w postaci dynamicznie łączonej biblioteki klas.

Struktura działającego systemu

OB!OK! jest biblioteką klas. Nie oznacza to dowolności w jej wykorzystywaniu. W momencie zainicjowania przez aplikację zostają utworzone zaprojektowane struktury systemu i każdy dostęp do metod obiektów tworzących *OB!OK!* jest kontrolowany pod względem poprawności odwołania i bezpieczeństwa pracy całego systemu. Schemat struktury działającego OSZBD *OB!OK!* przedstawia rys.1.



Rys. 1. Struktura działającego systemu *OB!OK!*

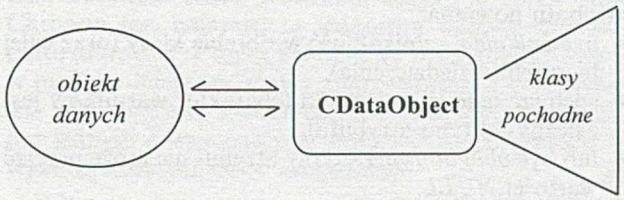
Prezentowana całość mieści się w schemacie systemu informacyjnego (tabela 1) na poziomie II: z jednej strony są dane zapisane fizycznie, z drugiej strony aplikacja.

Górna część schematu, obiekt **CDataWorld** reprezentujący pewien **świat danych** oraz jego obiekty pomocnicze zapewniają translację danych z postaci fizycznej do postaci abstrakcji definiowanej przez model obiektowy. Dolna część schematu, tzn. obiekty: **CDataWorkSpace**, **CDataObject** i **CIndex** to elementy programistycznego interfejsu aplikacji. Tylko rola ostatniego z nich jest jasna intuicyjnie. Umożliwia on dostęp do otwartego indeksu. Pozostałe obiekty wymagają kilku słów wyjaśnienia.

CDataObject – programistyczna emanacja

W większości OSZBD język programowania jest rozszerzany o elementy języka manipulacji danymi. W efekcie staje się możliwe zaistnienie obiektów danych jako bezpośrednich elementów języka. Postępowanie takie prowadzi jednak do ukształtowania się języka programowania o ściśle określonym zastosowaniu [3].

W jaki sposób, w takim razie, wprowadzić obiekty danych do abstrakcyjnej przestrzeni aplikacji, nie rozszerzając użytego języka?



Rys. 2. Reprezentacja obiektu danych przez jego programistyczną emanację

W *OB!OK!* osiąga się to przez mechanizm ścisłego powiązania obiektu danych z reprezentującym go obiektem języka *C++* klasy **CDataObject** (nazwa tej klasy nie ma tutaj większego znaczenia). Zilustrowano to na rys.2. Obiekt ten można określić, jako *programistyczną emanację* (czyli kopię, ale będącą odbiciem „prawdziwego istnienia"). Jest ona „ucieleśnieniem" abstrakcji **obiektu danych**, stanowi programistyczny interfejs umożliwiający dostęp do niego. Obiekt klasy **CDataObject** jest tylko tworzywem, wzorcem końcowej postaci narzędzia, które kształtujemy według informacji zapisanej w definicji *klasy danych* (pojęcie analogiczne do **obiektu danych**). Wszystkie struktury, opisujące poszczególne klasy danych, są przechowywane w jednym z plików świata danych i dostęp do nich jest jedynie możliwy poprzez obiekt klasy **CDataWorld**.

Podstawowym zadaniem klasy **CDataObject** jest dostarczenie elastycznego szkieletu będącego w stanie dostosować się do dowolnej zdefiniowanej klasy danych. Jej struktura umożliwia w szczególności przechowywanie stanu atrybutów o dowolnie skomplikowanych typach strukturalnych (np. lista, stos czy też tablica rzadka) oraz umożliwia obsługę wartości NULL.

Klasa **CDataObject** zapewnia implementację kilkudziesięciu predefiniowanych (np. zapewniających dostęp do stanu atrybutów) metod **obiektu danych** oraz umożliwia (poprzez dziedziczenie po niej innych klas) proste dodanie dalszych definiowanych przez użytkownika metod **obiektów danych**. Klasy dziedziczące po **CDataObject** mogą zmieniać działanie metod predefiniowanych.

Obiekt danych w systemie *OB!OK!* jest dostępny dla użytkownika (aplikacji) wtedy i tylko wtedy, gdy jest skojarzony z odpowiadającym mu obiektem klasy **CDataObject** (lub klasy pochodnej od niej). Sprzężenie to zapewnia oczywiście sam OSZBD. Znajduje to odbicie w kilku możliwych stanach życia obiektu danych. Ich specyfikację zawiera tabela 2.

Tabela. 2. Stany życia obiektu danych w systemie *OB!OK!*

stan	dysk	pamięć	opis
niebyt	-	-	obiekt nie istnieje
prenatalny	-	+	obiekt został utworzony i jest definiowany jego stan
aktywny	+	+	utworzony wcześniej obiekt został obudzony (wczytany do pamięci może być tylko odczytywany lub po zablokowaniu w stanie edycji
letarg	+	-	obiekt jest przechowywany na dysku, aplikacja nie ma bezpośredniego dostępu do jego metod

Obiekt klasy **CDataObject** jest zatem jedynie „buforem" dla pewnego **obiektu danych**. Uzgodnienie ich wzajemnej relacji jest dokonywane przez mechanizm synchronizacji stanu (przepisanie od dysku do pamięci lub odwrotnie).

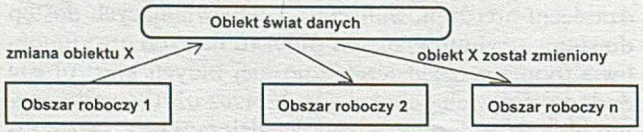
Obiekt klasy **CDataObject** zapewnia ponadto realizację kilku technicznych własności atrybutów, nie wynikających z modelu obiektowego. W klasie danych możemy dla każdego atrybutu określić jego dodatkowe cechy (mechanizm szeroko stosowany w SZBD) wiążące się z pojęciem integralności danych. Wartość dowolnego atrybutu powinna:

- *być unikalna* – unikalność w obrębie klasy (oraz całej hierarchii dziedziczenia),
- spełniać nałożone warunki (charakter warunków jest zależny od typu atrybutu),
- lub być *obligatoryjna* – dany atrybut nie może przyjąć wartości *NULL*.

CDataWorkSpace – obszar roboczy, realizacja modelu klient-serwer

W *OB!OK!* zaimplementowano mechanizmy architektury klient-serwer. Poprzez tzw. **obszar roboczy** przydziela się „okno robocze" każdemu zgłaszającemu się do systemu klientowi. Obszar roboczy, reprezentowany przez obiekt klasy **CDataWorkSpace**, izoluje poszczególnych klientów od siebie, zapewnia koordynację ich działań oraz dostęp do usług systemu (np. usług zarządzania obiektami danych). Obszar roboczy umożliwia między innymi selektywne filtrowanie informacji przechowywanej w bazie danych przez oddziaływanie na wszystkie narzędzia (np. otwarte indeksy), używane w tym obszarze. Jednym z argumentów niezbędnych do otwarcia obszaru roboczego jest rezultat pracy obiektu **CSearch** (obiekt **CSearch** realizuje wyszukiwanie w bazie na podstawie kryteriów sformułowanych w postaci specjalnej struktury wejściowej).

Otwartymi obszarami roboczymi zarządza obiekt **CDataWorld** (reprezentujący świat danych). Współdziałanie obszarów roboczych jest oparte o mechanizmy modelu klient-serwer oraz mechanizm przesyłania wiadomości (rys.3.), przy czym niektóre wiadomości są propagowane przez obszary robocze aż do aplikacji użytkownika.



Rys. 3. Schemat koordynacji obszarów roboczych w *OB! OK!*

Bardzo istotną rolę pełni ponadto mechanizm *blokowania* **obiektów danych**. Dany obiekt (identyfikowany jednoznacznie przez swój identyfikator) może być w danym momencie zablokowany tylko w jednej ze swoich *programistycznych emanacji*. Tylko w jednym miejscu, skojarzony z **obiektem danych** obiekt klasy **CDataObject** może być w stanie edycji.

Połączenie mechanizmu wiadomości z mechanizmem blokowania daje możliwość pełnej koordynacji pracy klientów. Przykładowo:

- o zmianie stanu dowolnego z obiektu danych w jednym z obszarów roboczych są powiadamiane przez wiadomości wszystkie pozostałe obszary robocze, w których dany obiekt jest używany,
- podczas kasowania, zanim obiekt zostanie usunięty z systemu, pozostałe obszary robocze mają szansę utworzyć nowy obiekt, który jest kopią kasowanego.

Do systemu został również włączony mechanizm *transakcji*. Posiada cechy obiektu, co daje dużo swobody projektantowi aplikacji docelowej. W założeniu transakcja miała zawierać wszystkie elementarne operacje związane z obsługą danego obiektu danych w czasie jego aktywności (np. od momentu obudzenia do uśpienia). Projektant aplikacji ma jednak możliwość powiązania operacji dotyczących kilku obiektów z jedną transakcją i traktowania ich dalej jako operacji.

* * *

Ciekawym doświadczeniem z pracy nad systemem *OB!OK!* jest to, że przy zaangażowaniu stosunkowo skromnych nakładów udało się zbudować obiektowy system zarządzania bazą danych o niezbyt wygórowanych wymaganiach sprzętowych. Jednocześnie możliwości, jakie oferuje ten system i efektywność jego pracy okazały się zadawalające. Sprawdzono w praktyce nieco odmienne podejście od zastosowanego w dotychczas zrealizowanych systemach OSZBD. Zaletą zbudowanego systemu jest to, że projektując system bazy danych z zastosowaniem *OB!OK!* możemy się skupić na projekcie aplikacji, a baza danych staje się jedynie „częścią" tej aplikacji. Językiem projektowania całego *systemu bazy danych* jest język programowania aplikacji. W trakcie prac nad zbudowaniem *OB!OK!* wyłoniło się ciekawe pojęcie **świat danych**, warte dalszego badania jego własności. Współpraca z projektantami i programistami systemu multimedialnego, wytworzonego w oparciu o system *OB!OK!*, wykazała, że informatycy posiadający doświadczenie w wykorzystywaniu technologii obiektowej łatwo akceptują użyty w *OB!OK!* model obiektowego systemu zarządzania bazą danych.

LITERATURA

[1] Delobel C., Lecluse C., Richard P.: Databases: From Relational to Object-Oriented Systems. International Thomson Publishing, 1995
[2] Lipski W.: On Databases with Incomplete Information. Journal of the ACM, 28(1), 1981
[3] Piasecki M.: Koncepcja modelu dla obiektowej bazy danych. Informatyka 7.1996.

W każdej chwili można zaprenumerować INFORMATYKĘ oraz kupić zaległe numery z bieżącego roku i lat ubiegłych.

Reprezentacja czasu w modelu konceptualnym SI

Krzysztof Kania
Jerzy Gołuchowski

Niezależnie od sporów naukowych nad naturą czasu trzeba przyjąć w modelu rzeczywistości takie jego ujęcie, które będzie najlepiej spełniać oczekiwania użytkowników i służyć rozwiązywanym przez nich problemom praktycznym. To pragmatyczne podejście sprawia, że informatyk musi znać możliwe modele czasu, aby dobrać najwłaściwszy dla danej dziedziny i potrzeb użytkownika.

Ze względu na wzrastające zapotrzebowanie na systemy uwzględniające historię zjawisk, konieczne stało się stworzenie metodyki budowy tej klasy systemów. Istniejące modele są dobrze zbadane, jak choćby modele OAR, ale są też i zupełnie nowe wykorzystujące nieklasyczne systemy logiki.

Ontologia czasu w SI

Próby wyjaśnienia struktury czasu i opisu jego cech są podejmowane przez przedstawicieli różnych dziedzin nauki: filozofii, fizyki, logiki, a ostatnio także informatyki (sztuczna inteligencja). Teorie głoszone przez nich bardzo często stoją w sprzeczności z pojęciem czasu, rozumianym intuicyjnie przez przeciętnego człowieka.

Systemy informatyczne muszą godzić naukowe i potoczne pojęcia czasu z jednej strony dla zapewnienia poprawności budowanych modeli, a z drugiej – prostoty rozumienia i komunikacji.

Do istotnych dla budowy SI rozstrzygnięć natury czasu należy [7,8]:

- model i sposób upływu czasu,
- sposób określania czasu,
- sposób ujęcia czasu w modelach danych.

Prace informatyków poświęcone modelowaniu czasu w zastosowaniach ekonomicznych koncentrują się obecnie wokół dwóch podstawowych typów modeli czasu: **liniowego i gałęziowego**. W pierwszym modelu kierunek czasu przebiega w sposób całkowicie uporządkowany od przeszłości ku przyszłości, natomiast w modelu gałęziowym, zwanym także modelem alternatywnych przyszłości, czas jest liniowy od przeszłości do chwili obecnej, gdzie może dzielić się na wiele linii, reprezentujących możliwe sekwencje przyszłych działań lub wartości, interesujących użytkownika. Tak więc w modelu gałęziowym czas przedstawia się jako drzewo, którego korzeń umiejscowiony jest w punkcie określającym teraźniejszość. Modele takie służyć mogą do konstruowania planów i tworzenia scenariuszy działań.

Nie są to jedyne postacie linii przebiegu czasu. W niektórych zastosowaniach mogą być również przydatne inne modele czasu np. model cykliczny służący reprezentacji zjawisk powtarzalnych (np. zmiana pór roku) lub rekurencyjnych.

Kwestia, która również wymaga rozstrzygnięcia w toku tworzenia SI (modelowania dziedziny przedmiotowej) dotyczy sposobu upływu czasu. Należy przyjąć model ciągłego lub dyskretnego przebiegu czasu.

W modelu ciągłym czas jest izomorficzny ze zbiorem liczb rzeczywistych. Model taki nie zawiera żadnych „dziur", a każda liczba rzeczywista odpowiada jednemu „punktowi" czasu.

W modelu dyskretnym zakłada się, że czas składa się z pewnych niepodzielnych jednostek o określonym trwaniu. Czas jest skwantowany, a zbiór jego jednostek jest równoliczny ze zbiorem liczb naturalnych. Podejście takie wymusza stworzenie jednostki czasu nazywanej momentem, kwantum czasu, jednostką czasu lub chrononem. Chronon jest najmniejszą jednostką upływu (trwania) czasu (duration of time), która może być reprezentowana w modelu. Niepusty, ciągły zbiór chrononów określany jest jako interwał. Dobór odpowiedniej jednostki czasu jest jednym z ważniejszych aspektów projektowania SI wykorzystujących dane temporalne.

Kilka aspektów praktycznych zadecydowało o częstszym zastosowaniu modelu dyskretnego [5]:

- implementacja (techniczna) modeli danych temporalnych wymaga dyskretnego opisu czasu,
- określenie czasu w pojęciu potocznym odpowiada interwałowi, np. 12:06 nie odpowiada punktowi, lecz raczej wskazuje na jednostkę (w tym wypadku minutę),
- dyskretny model czasu pozwala na naturalne modelowanie działań trwających w czasie,
- przyrządy pomiarowe są nieprecyzyjne.

Modelując dany fragment rzeczywistości należy także rozważyć sposób określania czasu otrzymywanej informacji. Wyróżnia się dwa podstawowe sposoby określania czasu, a w konsekwencji dwa podstawowe jego rodzaje [6]:

Czas **absolutny** – niezależny od przebiegu zjawiska i nie związany z jakimkolwiek punktem odniesienia, wiążący zdarzenie z konkretną wartością lub zbiorem wartości opisujących czas (kalendarzem). Przykładem takiego opisu są zdania:

Przyjęto dostawę do magazynu 12 marca 1994 roku.

Decyzję o zakupie podjęto 7 kwietnia 1994 roku o godzinie 10:15.

Czas **relatywny** – określający czas wystąpienia jednego zdarzenia poprzez określenie relacji z innym zdarzeniem. Czas relatywny występuje w dwu odmianach: **absolutnej** i **względnej**. Czas relatywny absolutny określa czas wystąpienia zjawiska poprzez określenie relacji z innym zdarzeniem oraz system kalendarza np.:

Pracownik X zwolnił się w 30 dni po przyjęciu do pracy.

Czas relatywny względny wskazuje jedynie na kolejność zdarzeń lub działań, nie wskazując na czas, który upłynął pomiędzy nimi, np.:

Zbankrutował po zawarciu umowy.

Z przedstawionymi pojęciami czasu absolutnego i względnego wiążą się pojęcia czasu zewnętrznego i wewnętrznego. Stanowią one ważny aspekt analizy informacji.

Jako czas zewnętrzny traktujemy czas stwierdzenia faktu. Na przykład zdaniu *„Przyjęto towar do magazynu"* wypowiedzianemu 15.03.1995 przypisujemy czas zewnętrzny wyznaczony przez datę wypowiedzi. Czas wewnętrzny to czas zawarty w samej informacji. Zdanie *„Przyjęto towar 2 marca 1995"* zawiera czas wewnętrzny oraz jest opisywane czasem zewnętrznym określającym, kiedy owo zdanie zostało wypowiedziane.

Podobnym ujęciem tego aspektu na poziomie modelu danych jest rozróżnienie między tzw. czasem **rzeczywistym** (valid time), a czasem **transakcyjnym** (transaction time). Czas rzeczywisty określa moment, w którym zdarzenie faktycznie zaistniało w otoczeniu, niezależnie od tego, czy informacja została gdziekolwiek przyjęta lub zapisana. Określenie to może dotyczyć także przyszłości, jeśli wiemy kiedy jakieś zdarzenie wystąpi. Czas transakcyjny to czas, w którym informacja o występujących zdarzeniach została zarejestrowana przez system. Od tego momentu baza danych systemu odzwierciedla stan istniejący. Idealnym stanem byłoby gdyby obydwa te czasy były takie same.

Dane temporalne wprowadzane do systemu mogą być czasowo niedookreślone, co oznacza „nie wiadomo dokładnie kiedy". Należy podkreślić, że nieostrość nie dotyczy faktu, że zdarzenie w ogóle wystąpiło (lub nastąpi), lecz momentu, w którym miało (lub będzie miało) miejsce.

Niedokładność informacji tego rodzaju może być konsekwencją [7]:

- różnej „ziarnistości" (granularity) czasu przyjętej w różnych systemach, a wskutek tego różnej dokładności pomiaru i rejestracji czasu; na przykład czas zapisany w systemie z dokładnością „do dnia" w systemie zapisującym również godziny i minuty, będzie zapisany nieprecyzyjnie,
- niepewności planowania – czas zdarzeń przyszłych często określany jest nieprecyzyjnie, np. od trzech do sześciu dni,
- nieprecyzyjności metod datowania, co dotyczy zwłaszcza informacji odległych historycznie.

Kolejnym problemem jest rejestracja i reprezentacja zdarzeń. **Zdarzeniem** (event) określa się każde zjawisko wewnątrz obiektu lub w jego otoczeniu, które powoduje konieczność dokonania zapisu w bazie tak, aby jej zawartość oddawała stan rzeczywisty zgodnie z potrzebami odbiorcy.

Zdarzenia podzielić można na **zewnętrzne** i **wewnętrzne.** Zdarzenia zewnętrzne powstają niezależnie od działań w obiekcie, np. nadejście zamówienia, przywiezienie towaru itp. Zdarzenia wewnętrzne – występujące w modelowanym obiekcie – podzielić można na homeostatyczne – związane z badaniem stanu równowagi w systemie oraz terminowe – których definicje związane są z czasem, a ich wystąpienia z jego upływem [1]. Zdarzenia nie mają trwania i odbywają się natychmiast. W zależności od przyjętego modelu można moment ich zajścia interpretować jako „punkt" lub chronon.

Przykład formalnego ujęcia czasu w modelu konceptualnym

Prace nad formalnym ujęciem czasu w modelach konceptualnych zaowocowały opracowaniem różnych podejść metodologicznych do tworzenia SI, ujmujących w sposób jawny dynamikę dziedziny przedmiotowej. Różnią się one przyjętą ontologią, zastosowanym formalizmem oraz sposobem ujęcia czasu w danych. Omówienie oraz porównanie wybranych metodologii można znaleźć między innymi w pracach [3,7,8].

Przykładową metodą formalnego ujęcia czasu w modelowaniu konceptualnym jest metoda Conceptual Information Model (CIM) [8]. Dla prezentacji użyteczności proponowanego podejścia do definiowania systemu posłużymy się opisem działalności hurtowni.

Podstawowym rozstrzygnięciem w trakcie modelowania jest określenie wielkości chrononu, na którym będzie opierać się zapis wartości danych temporalnych w modelu. Dla większości zastosowań ekonomicznych wystarczający jest zbiór wartości {rok:miesiąc:dzień}, chociaż istnieją dziedziny przedmiotowe, w których tak zdefiniowany okres jest zbyt długi (dwa zdarzenia w tym samym interwale są traktowane jako równoczesne) i wymaga skrócenia. W tym przykładzie chrononem jest dzień określany za pomocą zwykłej daty.

Pierwszym krokiem w modelowaniu jest, podobnie jak w klasycznym podejściu, wyodrębnienie i opisanie obiektów występujących w rzeczywistości:

OBIEKTY: Towar, Dostawca, Odbiorca, Zamówienie,..

Podobnie dokonywany jest również opis atrybutów, z tym że atrybuty temporalne są dodatkowo oznaczane.

```
obiekt: Towar
atrybuty:
    id_towaru:        string;
    ilość_towaru(t): real+;
    cena_towaru(t): real+;
koniec Towar
```

```
obiekt: Dostawca
atrybuty:
    id_dostawcy:    string;        •••
    adres(t):       string;
koniec Dostawca
```

Kolejnym krokiem jest określenie zdarzeń związanych z definiowanymi obiektami, które mogą wpłynąć na zmianę wartości atrybutów. Opis zdarzenia obejmuje wskazanie obiektów, atrybutów zdarzenia oraz rodzaju wystąpienia (wewnętrzne lub zewnętrzne).

```
ZDARZENIA: Przyjęcie_towaru, Sprzedaż_towaru, Zmiana_ceny, ....

zdarzenie: Przyjęcie_towaru
    towar:        id_towaru;
    ilość:        ilość_towaru;
    data_p:       data;
    cena_zak:     cena_towaru;
    dostawca:     id_dostawcy;
    ident:        data_p, towar, dostawca;
wystąpienie:      zewnętrzne

zdarzenie: Sprzedaż_towaru
    towar:        id_towaru;
    ilość:        ilość_towaru;
    data_s:       data;
    cena_sprz:    cena_towaru;
    odbiorca:     id_odbiorcy;
    ident:        data_s,towar,odbiorca;
wystąpienie:      zewnętrzne

zdarzenie: Zmiana_ceny
    towar:        id_towaru;
    cena:         cena_towaru;
    data_z:       data;
    ident:        towar, data_z;
wystąpienie:      wewnętrzne

                  •••
```

Następnym etapem definiowania modelu jest konstrukcja reguł integralności. Dzięki nim możliwe jest odwzorowanie połączeń pomiędzy zmianami wartości atrybutów – oraz obliczenie ich kolejnych i bieżących wartości – a informacją o zdarzeniach, które zaszły lub które dopiero zajdą. Określone w języku naturalnym reguły są zapisywane przy pomocy wyrażeń logicznych. W rozpatrywanym przykładzie dla obiektu Towar reguły zapisane w języku naturalnym mogą być sformułowane następująco:

Ilość towaru w danej chwili jest równa sumie ilości przyjętych w zdarzeniach Przyjęcie_towaru i pomniejszonej o ilości wydane w zdarzeniach Sprzedaż_towaru dotyczących danego towaru, które nastąpiły przed daną chwilą.

Cena towaru w danej chwili jest ustalona przez Cenę_zak z ostatniego zdarzenia Przyjęcie_towaru dla danego towaru powiększonej o Marżę lub jeśli po przyjęciu nastąpiła Zmiana_ceny to przez Cenę ustaloną w tym zdarzeniu.

Formalny zapis odpowiednich reguł dla obiektu Towar wygląda następująco:

```
obiekt Towar
    atrybuty:
    id_towaru:              string;
    ilość_towaru(t):        real+;
                reguła:
            ∀id_towaru( ∃id_towaru(Przyjęcie_towaru(id_towaru)=id_towaru
                    ∧ Przyjęcie_towaru(data_p)≤Data_dziś)
                    ∧ ∃id_towaru(Sprzedaż_towaru(id_towaru)=id_towaru
                    ∧ Sprzedaż_towaru(data_s)≤Data_dziś))→
            →ilość_towaru=(∑Przyjęcie_towaru(ilość)-∑Sprzedaż_towaru(ilość));
    cena_towaru(t):             real+;
                reguła:
            ∀id_towaru[
                    ((∃id_towaru(Przyjęcie_towaru(id_towaru)=id_towaru)
                    ∧∃id_towaru(Zmiana_ceny(id_towaru)=id_towaru)
                    ∧max(Przyjęcie_towaru(data_p)>max(Zmiana_ceny(data_z))→
            →cena_towaru=Przyjęcie_towaru(cena)*Marża )
            ⊕       /* exclusive OR */
                    ((∃id_towaru(Przyjęcie_towaru(id_towaru)=id_towaru)
                    ∧¬∃id_towaru(Zmiana_ceny(id_towaru)=id_towaru)→
            →cena_towaru=Przyjęcie_towaru(cena)*Marża )
            ⊕
                    ((∃id_towaru(Przyjęcie_towaru(id_towaru)=id_towaru)
                    ∧∃id_towaru(Zmiana_ceny(id_towaru)=id_towaru)
                    ∧max(Przyjęcie_towaru(data_p))≤max(Zmiana_ceny(data_z))→
            →cena_towaru=Zmiana_ceny(cena))];
koniec Towar.
```

Wskazać należy, że jeśli nie dysponujemy systemem zarządzania danymi temporalnymi, zdefiniowane reguły mogą być podstawą lub wskazówką do konstrukcji konkretnych procedur tworzonych na etapie implementacji. W aktywnych SBD mogą być obsługiwane bezpośrednio przez moduł zarządzania regułami.

Określić można również tzw. ograniczenia (więzy) globalne (global constrains), które regulują ogólne warunki zajścia poszczególnych zdarzeń np.

OGRANICZENIA GLOBALNE:

```
Ilość towaru sprzedanego w jednym zdarzeniu >5
∀id_towaru ∀id_ odbiorcy∀data_s(ilość_towaru>5)
```

Zauważmy, że w modelu tym podkreślona została zmienność w czasie wybranych atrybutów, wskazane zostały warunki zmiany wartości i ich powiązania ze zdarzeniami. Dynamika zjawisk ma także swoje odbicie w przyjętych sposobach wyprowadzania nowych wartości atrybutów.

Praktyczne możliwości realizacji koncepcji systemu temporalnych baz danych

Temporalne modele baz danych, stanowiące techniczną realizację przedstawionych idei, powinny być zdolne do odzwierciedlania dynamicznej natury przedstawianej rzeczywistości. Aby było możliwe wykorzystanie takich modeli spełnione muszą być trzy podstawowe wymagania [6]:

- Można uchwycić i interpretować temporalny charakter danych.
- Należy dane zapisywać tak, by zachować ich czasowy kontekst.
- Należy dysponować mechanizmem odzyskiwania takich danych, bądź to przez bezpośrednie pytania, bądź przez mechanizmy indukcyjno-dedukcyjne.

Jest oczywiste, że aby można było uczynić zadość tym wymaganiom bazy danych powinny zostać skonstruowane inaczej niż dotychczas. Dane temporalne mogą być traktowane tak jak wszystkie inne dane lub wyróżniane specjalnym sposobem zapisu.

Problem pojawia się wtedy, gdy przedmiotem zapytań stają się:

- czas zapisu w bazie danych,
- czas w modelowanym świecie,
- wzajemne interakcje pomiędzy systemem a realnym światem.

W konsekwencji przestaje wystarczać to, że czas jest jedną z wielu danych i musi stać się dodatkowym, podstawowym aspektem procesu ewolucji danych oraz samej bazy danych.

Najwcześniejszym opracowaniem na temat zapisu informacji temporalnej w BD była praca [2], w której Ben-Zvi wykazał możliwość rozbudowy tradycyjnego modelu relacyjnego do postaci wystarczającej dla przedstawienia danych w ich pełnym czasowym ujęciu. Wykazał on, że dodanie do tradycyjnych relacji dodatkowych atrybutów wystarcza do udzielenia odpowiedzi na wszystkie pytania dotyczące zmienności wartości danych w czasie.

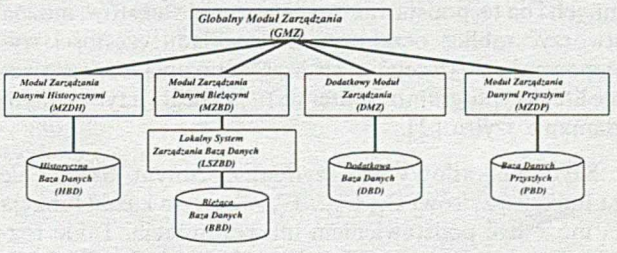
Biorąc pod uwagę ujęcie różnych wymiarów czasu w bazie, można wyróżnić następujące rodzaje temporalnych baz danych:

- historyczne bazy danych (historical databases),
- transakcyjne bazy danych (rollback databases, transaction databases)
- (bi)temporalne bazy danych (temporal databases, bitemporal databases)

W literaturze przedmiotu można spotkać różne definicje wymienionych teminów. Niektórzy autorzy używają terminu *temporal databases* w szerokim zakresie – dla wszystkich typów baz wspomagających manipulacje danymi temporalnymi, inni rezerwują go dla baz łączących możliwości historycznych i transakcyjnych baz danych.

Temporalne bazy danych wymagają innego SZBD niż tradycyjne bazy, które mogą być traktowane jedynie jako część SI modelującego dynamiczną rzeczywistość. Wskutek zredukowania reprezentacji czasu w BD do roli atrybutu, tradycyjne SZBD mogły być z tego punktu widzenia stosunkowo proste. Zadanie SZBD ograniczono do kontroli formalnej poprawności wprowadzanych określeń czasu.

Niestety, prostota konstrukcji stosowanych obecnie SZBD spowodowała znaczą komplikację ich zastosowań, zwiększoną pracochłonność tworzenia modeli dynamicznych dziedzin przedmiotowych oraz niską wydajność programistów. Stąd dostrzeżono potrzebę rozbudowy SZBD i poszukiwania innej, bardziej odpowiedniej ich architektury.



Rys. Architektura systemu temporalnych baz danych. Źródło: [3]

dokończenie na s.16

Metody szyfrowania informacji

Naceur Frih
Politechnika Śląska
Instytut Informatyki

Już w najdawniejszych czasach dla bezpiecznego przesyłania informacji używano szyfrów. Podobnie, jak i dzisiaj, celem szyfrowania było uniemożliwienie odczytania informacji, przechwyconej przez niepowołane osoby. W tradycyjnym szyfrowaniu stosowano przestawienia znaków w tekście pisanym i podstawienia pseudoznaków pod dane litery. W celu zwiększenia stopnia trudności odczytu do szyfrowania stosowano długie klucze.

Obecnie oprócz tradycyjnych metod szyfrowania istnieją nowe metody, których stosowanie stało się możliwe dzięki dużym szybkościom współczesnych komputerów [5]. Informacje komputerowe można szyfrować zarówno sprzętowo, jak i programowo. Obecnie dąży się do stworzenia tak złożonego i zawikłanego algorytmu szyfrowania, aby nawet kryptolog, otrzymujący dowolnie wybrane obszerne fragmenty zaszyfrowanego tekstu, nie mógł ich wykorzystać.
Najczęściej spotykanymi tradycyjnymi metodami szyfrowania są:

- szyfrowanie **przestawieniowe**,
- szyfrowanie **podstawieniowe**,
- szyfry **kaskadowe**.

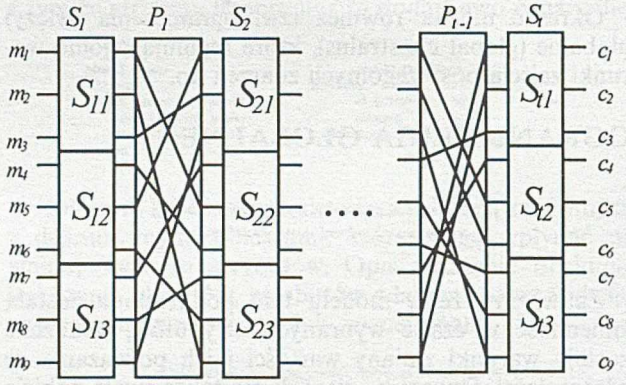
Szyfrowanie **przestawieniowe** jest wczesną metodą używaną na wiele stuleci przed pojawieniem się komputerów. Przykładem bardzo prostego szyfru jest szyfr Cezara. Był on przypuszczalnie używany przez Juliusza Cezara [6]. Jest to prosty szyfr przestawieniowy, w którym każdą literę alfabetu zastępuje się inną, otrzymaną w wyniku przesunięcia litery źródłowej o K pozycji alfabetu. Alfabet tworzy cykl zamknięty to znaczy po literze Z następuje litera A. Juliusz Cezar stosował ten szyfr dla $K = 3$.

Istnieją cztery typy szyfrów **podstawieniowych**: monoalfabetyczne, homofoniczne, wieloalfabetyczne i poligramowe. Zasada ich działania opiera się na zastąpieniu bitów, znaków lub bloków znaków ich ustalonymi zamiennikami. Metoda porównania częstości występowania liter w kryptogramie z częstościami oczekiwanymi (np. w języku polskim jedne litery występują częściej od innych i na tej podstawie, analizując wiele tekstów, można stworzyć tablicę oczekiwanego rozkładu częstości występowania poszczególnych liter) umożliwia dopasowanie liter kryptogramu do liter testu jawnego, a tym samym złamanie szyfru [2].

Szyfr **kaskadowy** jest szyfrem złożonym. Składa się z t funkcji (szyfrów) $F_1, F_2, \ldots F_t$, w którym każda funkcja F_i może być podstawieniem lub permutacją. Takie rozwiązanie przyjęto w maszynie szyfrującej „Lucifer", zaprojektowanej przez Feistela [3] z firmy IBM.

LUCIFER

W sieciach komputerowych zarówno szyfry przestawieniowe, jak i podstawieniowe, okazały się niewystarczające. Stworzono nowy typ szyfrów kaskadowych, opierający się na złożeniu t funkcji $F_1, \ldots F_t$, w którym każda funkcja F_i może być podstawieniem lub permutacją. Dzięki temu uzyskuje się bardzo dobre rozprowadzenie informacji z tekstu jawnego po całej dziedzinie możliwych informacji zaszyfrowanych. Taką koncepcję przyjęto przy tworzeniu urządzenia szyfrującego **Lucifer** firmy IBM, gdzie zastosowano na przemian przestawienia i podstawienia. Na rys.1 przedstawiono zasadę działania algorytmu dla bloku 9-bitowego. W praktyce algorytm operował na większych blokach. Szyfr składa się z wykonywanych na przemian podstawień S_i oraz permutacji (przestawień) P_i, co można wyrazić wzorem: $C = E_k(M) = St\ oP_{t-1} o \ldots oS_2\ oP_1\ oS_1(M)$, przy czym każde podstawienie S_i jest funkcją klucza k. Układ podstawiający S_i podzielono na 3 mniejsze układy $S_{i1} \ldots S_{i3}$ działające na 3-bitowych fragmentach bloku, co ma na celu uproszczenie układów mikroelektronicznych [8].



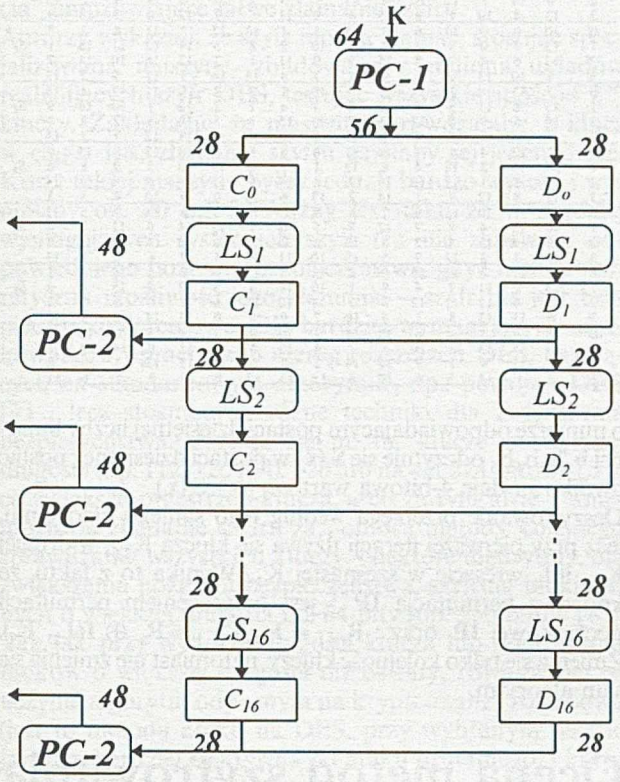
Rys. 1. Szyfr podstawieniowo-permutacyjny

Szyfr DES (Data Encryption Standard)

Na bazie szyfru Lucifer został opracowany standard szyfrowania danych DES (Data Encryption Standard), służący początkowo do transmisji informacji transmitowanych przez agendy rządowe USA. Wprowadziło go Narodowe Biuro Normalizacji w 1977 roku. Kompletny algorytm jest przedstawiony w postaci schematu na rysunkach 2÷4.

Kluczem jest tu 64-bitowy ciąg (blok) zerojedynkowy k, składający się z ośmiu bajtów, w których ósmy bit jest bitem parzystości: przyjęto, że każdy bajt zawiera nieparzystą liczbę jedynek. Klucz K służy do wytworzenia, zgodnie z zasadą pokazaną na rysunku 2, 48-bitowych kluczy

szyfrujących k_1, k_2,...,k_{16}. Na wstępie ciąg wejściowy k zostaje podawany do **PC-1** i następnie podzielony na dwa bloki C_0 i D_0 według modelu przedstawionego w tabeli 1. W kolejnych krokach i (i = 1,2,...,16) bloki C_i i D_i uzyskuje się wskutek cyklicznego przesunięcia w lewo LSi (i = = 1,2,...,16) odpowiednio, bloku C_{i-1} i D_{i-1} (tab.2.).



Rys. 2. Sposób tworzenia klucza

Klucz k_i uzyskuje się dokonując permutowanego wyboru PC-2 zdefiniowanych według zasad w tabeli 3. Klucze pomocnicze k_1, k_2,...,k_{16} są wykorzystywane w procesie szyfrowania 64-bitowej wiadomości jawnej T (rys.3).

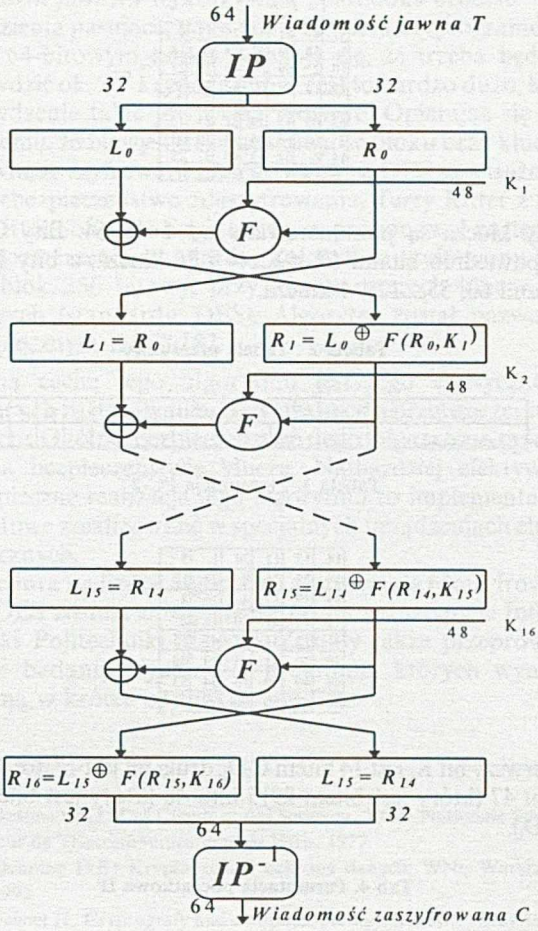
Po dokonaniu permutacji początkowej **IP** (tab. 4) są tworzone dwa bloki L_0 i R_0, a następnie w szesnastu krokach są dokonywane przekształcenia

$$L_i = R_{i--1}; \text{ oraz } R_i = L_{i--1} \oplus f(R_{i--1}, K_i),$$

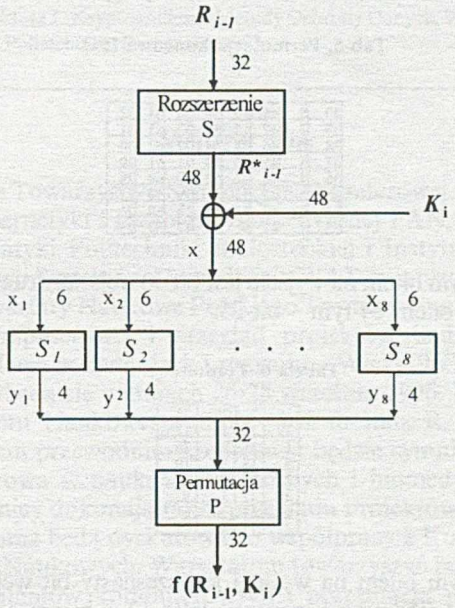
przy czym $\oplus$ oznacza sumowania modulo dwa odpowiednich bitów sumowanych bloków. Wartości funkcji f zależnej od R_{i-1} i klucza pomocniczego K_i wyznacza się zgodnie z zasadą przedstawioną na rys. 4 (liczby umieszczone na rysunkach oznaczają długość przesyłanego słowa) i w tabelach 4 i 5. Rozszerzenie bloku R_{i--1} do słowa 48-bitowego R^*_{i--1} uzyskuje się w taki sposób, że jeśli $R_{i--1} = b_1b_2...b_{32}$, to $R^*_{i-1} = b_{32}b_1b_2b_3b_4b_5b_6b_7b_8b_9b_8 b_9b_{10}b_{11}b_{12}b_{13}...b_{28}b_{29}b_{30}b_{31}b_{32}b_1$.

Operacja $\oplus$ przedstawiona na rys. 4, jest operacją sumowania modulo 2 odpowiednich bitów. Uzyskany 48-bitowy blok jest dzielony na osiem 6-bitowych bloków $X_1, X_2 ... X_8$.

Zastosowanie funkcji wyboru S_i, określonej w tabeli 7 do wejścia X_i daje w wyniku 4-bitowy blok wyściowy Y_i. Kolejne bloki Y_i są konkatenowane w 32-bitowe słowo $Y_1Y_2 ...Y_8$, podawane następnie permutacji P określonej w tabeli 6. Po uzyskaniu bloków L_{16} i R_{16} skonkatenowany 64-bitowy blok L_{16} R_{16} poddaje się permutacji końcowej (tabela 5) uzyskując 64-bitową wiadomość zaszyfrowaną C.



Rys.3. Algorytm szyfrujący DES



Rys. 4. Sposób wyznaczania wartości *funkcji f*

Tab 1. Permutacja PC-1

57	49	41	33	25	17	9	
1	58	50	42	34	26	18	C_0
10	2	59	51	43	35	27	
19	11	3	60	52	44	36	
63	65	47	39	31	23	15	
7	62	54	46	38	30	22	D_0
14	6	61	53	45	37	29	
21	13	5	28	20	12	4	

Bity klucza są ponumerowane od 1 do 64. Bity C_0 są odpowiednio bitami 57, 49,....44 i 36 klucza, a bity D_0 są bitami 63, 55,....12 i 4 klucza.

Tabela 2 . Tabela przesunięć

Krok i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Liczba przesunięć	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tabela 3. Permutacja PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Pierwszy bit K_i jest 14 bitem C_iD_i, drugi bit jest 17 itd.– aż do bitu 47 (który jest bitem 29) i bitu 48 (który jest bitem 32 C_iD_i).

Tab 4. Permutacja początkowa IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	25	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Pierwszym bitem na wyjściu jest bit 58 wejścia, drugim – bit 50 itd., a bit 64-tym – bit 7.

Tab 5. Permutacja koncowa IP-1

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Pierwszym bitem na wyjściu jest bit 40 wejścia, drugim – bit 8 itd., a bitem 64-tym – bit 25.

Tabela 6. Permutacja P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Pierwszym bitem na wyjściu jest szesnasty bit wejściowy, drugim – siódmy itd., wreszcie jako 32-gi bierze się bit 25. Jeśli $x_i = b_1b_2b_3b_4b_5b_6$, to wartość $S_i(x_i)$ wyznacza się w taki sposób, że na przecięciu wiersza o numerze odpowiadającym postaci dziesiętnej liczby binarnej b_1b_6 i kolumny

Tabela 7.Funkcje wyboru, $S_1, S_2, ..., S_8$

	Kolumna															
Wiersz	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S_1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S_2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S_3
2	13	6	4	9	8	15	3	0	11	1	2	12	15	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S_4
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	8	13	1	5	0	15	10	3	9	8	6	S_5
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S_6
2	9	14	15	5	2	8	12	3	7	0	4	10	4	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S_7
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S_8
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

o numerze odpowiadającym postaci dziesiętnej liczby binarnej $b_2b_3b_4b_5$ odczytuje się $S_i(x_i)$ w postaci dziesiętnej; postać dwójkowa daje 4-bitową wartość $y_i = S_i(x_i)$.

Deszyfrowanie przebiega według tego samego algorytmu, lecz przy pierwszej iteracji używa się klucza K_{16}, w drugiej K_{15} itd., wreszcie w szesnastej K_1, Wynika to z faktu, że końcowa permutacja IP^{-1} jest odwróceniem permutacji początkowej IP, oraz : $R_{i-1} = L_i$, $L_{i-1} = R_i \oplus f(L_i, K_i)$. Zmienia się tylko kolejność kluczy, natomiast nie zmienia się sam algorytm.

Ocena metod szyfrowania

Tradycyjne metody kryptograficzne, mimo że pozwalają na zwiększenie bezpieczeństwa danych w sieciach komputerowych, nie gwarantują całkowitej ochrony przed złamaniem szyfru. Podstawową ich wadą w stosunku do szyfru DES jest łatwość ich złamania, co ilustrują przedstawione poniżej przykłady. Stworzenie bezpiecznej metody szyfrowania nie jest proste.

Szyfry przestawieniowe łamie się metodą anagramową. Polega ona na szukaniu takich permutacji, przy których częstość występowania kombinacji dwuliterowych (digramów) jest zbliżona do odpowiedniej częstości w języku naturalnym (np. w języku angielskim najczęściej występują digramy TH i HE). Podobnie można postąpić z trigramami (kombinacje trzyliterowe).

Szyfry podstawieniowe łamie się przy pomocy analizy częstości występowania znaków. Najczęściej występujący w tekście zaszyfrowanym symbol odpowiada najczęściej występującemu znakowi języka naturalnego.

Opracowano algorytm relaksacyjny do przełamywania szyfrów podstawieniowych. Dla każdej litery *a* tekstu jawnego oraz litery *b* kryptogramu oblicza się początkowe prawdopodobieństwo tego, że litera *a* jest zaszyfrowana jako *b*. Prawdopodobieństwo to jest następnie iteracyjnie modyfikowane na podstawie częstości występowania trigramów.

Sytuacja jest podobna dla szyfrów homofonicznych z dłuższym alfabetem tekstu zaszyfrowanego. Jeśli alfabet ten jest dłuższy od tekstu szyfrowanego, szyfr może być nieprzełamywalny.

W 1976 r. (przed wprowadzeniem szyfru DES jako

standardu) słabymi stronami algorytmu zajmowali się Morris, Sloane i Wyner. Wskazali oni na dwie słabe strony algorytmu:
1. Długość klucza: klucz 56 bitowy nie zapewnia odpowiedniego bezpieczeństwa.
2. Skrzynki S: skrzynki te mogą posiadać „tajne przejścia" umożliwiające łatwe złamanie szyfru.
Autorzy wykazali, że szyfr można złamać, stosując specjalizowaną maszynę, zbudowaną z miliona układów realizujących szyfr DES, testując wszystkie spośród 2^{56} kluczy. Zakładając, że maszyna sprawdzałaby 1 klucz w ciągu 1μs, złamanie szyfru zajęłoby jej jeden dzień. Koszt takiej maszyny byłby jednak bardzo wysoki i wyniósłby ok. 20 ml $. Ważny jest fakt, że w bardziej wymagających systemach szyfr ten nie zapewnia odpowiedniego poziomu bezpieczeństwa, gdyż istnieje teoretyczna możliwość jego złamania – szyfr nie jest bezpieczny obliczeniowo. Dla bardziej wymagających użytkowników opracowano szereg rozszerzeń DES, bazujących na standardowym algorytmie, np.: potrójny **DES** [7] , lecz stosujących różne techniki dla zwiększenia bezpieczeństwa. Stosuje się m. in. klucze o większej długości (np. 112, 120 [7]), 128 lub nawet 224-bitowe [8]), co zwiększa przestrzeń klucza, a co za tym idzie – uniemożliwia złamanie szyfru za pomocą metody kolejnego sprawdzania wszystkich kluczy. Niektóre algorytmy, dla zwiększenia poziomu bezpieczeństwa, szyfrują bloki danych o większej długości niż 64-bity (np. 256 bitów [8]). Tak jak przy wyborze długości klucza lub szyfrowania bloków o większej długości niż 64-bity, IBM zamierzał uczynić algorytm odpornym na kryptoanalizę różnicową (jest to metoda ataku na DES, przy wybranym tekście jawnym bardziej efektywna od ataku brutalnego) i kryptoanalizę liniową (jest to metoda ataku na DES, przy znanym tekście jawnym) [9].

* * *

Informacje w sieciach komputerowych można szyfrować zarówno sprzętowo, jak i programowo. Algorytm szyfrowania typu DES, stworzony przez NBS (National Bureau of Standards), jest zaimplementowany w wersji programowej i sprzętowej. Jednym ze słabych punktów algorytmu DES jest długość szyfrowanego bloku. Niektóre *ataki z tekstem jawnym* wykorzystują „paradoks urodzin" do znalezienia pasującej pary, spośród wielu kryptogramów. Przy 64-bitowym bloku oczekuje się, że trzeba będzie sprawdzić ok. 2^{32} krypogramów. Jest to bardzo dużo, lecz sprawdzenie takie jest wciąż możliwe. Opierając się na założeniu, że niewystarczająca długość bloku oraz klucza sprawia, że szyfrowane dane za pomocą DES są narażone na niebezpieczeństwo zdeszyfrowania, Terry Ritter z firmy Ritter Software Engineering, zaproponował następujące rozszerzenie standardu. Założył, że szyfrowany będzie blok 256 bitowy, przy użyciu czterech kluczy 56 bitowych (standardu DES). Algorytm został nazwany „Bezpiecznym DES" [8].

Ważną cechą tego algorytmu jest jego elastyczność w różnych zastosowaniach związanych z przetwarzaniem danych. Ogólne bezpieczeństwo tego algorytmu opiera się na bezpieczeństwie klucza. Najbardziej efektywne i bezpieczne realizacje tego algorytmu to implementacje sprzętowe zrealizowane w specjalnych urządzeniach elektronicznych.
Procedura realizacji algorytmu szyfrowania (deszyfrowania) DES została zaimplementowana w Instytucie Informatyki Politechniki Śląskiej. Zostały także przeprowadzone badania wydajności programu, których wyniki zostaną w krótce opublikowane.

LITERATURA

[1] Beaucourt D.: Des Clients et des Serveurs., Ecole Nationale Superieur de Telecommunications de Paris. 1977
[2] Denning D.E.: Kryptografia i ochrona danych; WNt, Warszawa 1992
[3] Feistel H., Cryptografy and computer privacy. Sci. AM., May 1973, vol. 228(5), s. 15–23
[4] Frih N.: Bezpieczeństwo sieci komputerowych; ZN Pol.Śl. s. Informatyka, z. 28, Gliwice 1995
[5] Hoffman L. J.: Modern Methods for Computer Security and Privacy., Printice-Hall, Inc New Jersy 7632, USA 1977
[6] Kahn D.: The codebreakers. New. York, Macmillan co. 1967
[7] Riordan M. *„DESX"* Internet Newsgroup sci.crypt 1994
[8] Ritter T. *„Fenced DES"* Internet Newsgroup sci.crypt Texas 1994
[9] Schneier B.: Kryptografia dla Praktyków; WNT, Warszawa 1995
[10] Stokłosa J.: Kryptograficzne Metody Ochrony Danych, Wydawnictwo Politechnika Poznańska 1992.

Konferencja – systemy informatyczne rachunkowości

Komisja Rady Naukowej Stowarzyszenia Księgowych w Polsce i Oddział Okręgowy Stowarzyszenia Księgowych w Polsce organizują konferencję na temat atestacji i normowania systemów informatycznych rachunkowości.
Konferencja odbędzie się w Kołobrzegu, 23 września 1996 r. Zostaną przedstawione wstępne wyniki prac Zespołu Roboczego w Polsce i powołanego do opracowania wymagań funkcjonalnych i eksploatacyjnych wobec systemów informatycznych rachunkowości. Organizatorzy konferencji oczekują na propozycje referatów oraz ew. prezentacje systemów z dziedziny rachunkowości.
W sprawach organizacyjnych należy kontaktować się z p. dyr. Ireną Chylewską (tel. 0-91-339-832), a w sprawach merytorycznych korespondencję można kierować na e-mail: idziedz@uoo.univ.szczecin.pl**(ap)**

Wigry '96

Polskie Towarzystwo Symulacji Komputerowej, Instytut Biocybernetyki i Inżynierii Biomedycznej PAN, Instytut Informatyki Politechniki Białostockiej i Instytut Automatyzacji Systemów Dowodzenia WAT organizują Trzecie Warsztaty Naukowe Polskiego Towarzystwa Symulacji Komputerowej i przegląd projektów badawczych „Symulacja w badaniach i rozwoju – Wigry '96". Warsztaty odbędą się w dniach 26-28 września 1996. Na czele Komitetu Naukowego stanie prof. dr hab. R. Bogacz. Tematem przewodnim konferencji będzie symulacja komputerowa w naukach technicznych i biomedycznych. Uczestnicy dokonają także przeglądu projektów badawczych oraz będą dyskutować o współpracy z Komitetem Badań Naukowych. Warsztatom towarzyszyć będzie pokaz systemów komputerowych i prezentacja pakietów symulacyjnych. Informacje o warunkach uczestnictwa można uzyskać pod adresem: III Warsztaty Naukowe PTSK – Wigry '96 c/o IPPT PAN, ul.Świętokrzyska 11/21 00-049 Warszawa lub e– mail: mrejmund@ippt.gov.pl**(ap)**

Programowanie obiektowe w CA-Clipper

Marek Miłosz
Katedra Informatyki
Bogdan Wit
Katedra Zarządzania
Politechniki Lubelskiej

Ewolucja języków programowania jest naturalnym procesem. Wynika ona ze wzrostu wymagań stawianych przez użytkowników oprogramowaniu i konieczności sprostania im przez programistów. Przemiany wiążą się z powszechnością, dostępnością, lawinowym wzrostem mocy obliczeniowej komputerów przy znacznym obniżeniu kosztów. Naturalny w tym przypadku pęd ku pisaniu coraz większych i bardziej złożonych programów w coraz krótszym czasie był ograniczany przez niedoskonałość i nieprzystosowanie istniejących języków programowania. Doprowadził do powstania (a właściwie powrotu do starych idei) nowych mechanizmów, m.in. do programowania obiektowego.

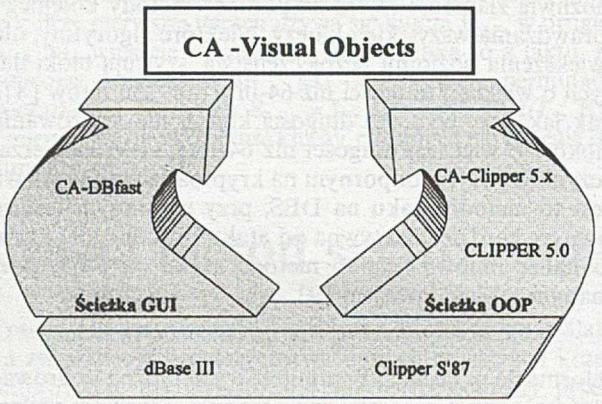
Programowanie zorientowane obiektowo (*Object Oriented Programming* – OOP) to sposób implementacji systemów, który zapewnia dużą efektywność, poprawność i odporność na błędy. OOP opiera się na kilku koncepcjach, określających metodę tworzenia struktury programu i zależności między jego modułami. Koncepcje te są niezależne od języka programowania.

OOP pozwala programistom na zmianę sposobu myślenia o rozwiązaniu problemów. Ponieważ obiekty same zawierają wiedzę o ich wewnętrznej strukturze, więc rozwój aplikacji i dodawanie nowych elementów nie wymaga przepisywania kodu. OOP upraszcza kod konieczny do rozwiązania problemu, dążąc do tego, by stał się on intuicyjny i sam prowadził do wyższego poziomu programowania. Różne języki programowania realizują obiektowe podejście programowania w różny sposób.

Jeden z najbardziej popularnych w Polsce języków programowania systemów zarządzania bazami danych Clipper (Computer Associates International) migruje skutecznie w stronę orientacji obiektowej.

Programowanie zorientowane obiektowo w języku CA-Clipper

Język programowania baz danych Clipper ulegał stopniowej ewolucji od znaczącej (i bardzo popularnej w swoim czasie w Polsce) wersji *SUMMER-87* (kompilator podzbioru języka programowania systemu dBase) do samodzielnego języka programowania systemów zarządzania bazami danych *CA-Clipper 5.3* i *CA-Visual Objects*. Rozwój programu Clipper podąża dwoma nie wykluczającymi się ścieżkami (rys. 1.). Jedna ścieżka związana jest z graficzną wymianą informacji (*Graphics User Interface*, GUI), druga z programowaniem obiektowym (OOP). Przykładem w rozwijaniu techniki graficznej wymiany informacji i dostosowanie do programu MS Windows jest CA-dBFast. Kierunek ten pozwala na sterowanie zdarzeniami i jest przystosowany do wykorzystania na różnych platformach (np. DOS, Windows, Unix). Drugi kierunek jest związany z rozwojem technik programowania obiektowego oraz architekturą klient/serwer i RDD (*Replacable Database Drivers*), pozwalającej na pracę z różnymi bazami danych. Wynikiem połączenia tych dwóch kierunków jest program CA-Visual Objects.



Rys. 1. Kierunki rozwoju języka programowania CA-Clipper

W najnowszej wersji języka CA-Clipper 5.3 nastąpiło połączenie zalet systemu operacyjnego DOS i systemu Windows. Wersja ta umożliwia projektowanie aplikacji w środowisku Windows dla środowiska DOS.

Zmiany w programie CA-Clipper czynią z niego coraz silniejsze narzędzie programowania obiektowego. Pomimo że CA-Clipper 5.3 nie jest jeszcze **prawdziwym językiem programowania obiektowego**, to zawiera konstrukcje językowe, które umożliwiają programowanie w sposób obiektowy.

Przez pojęcie prawdziwego języka obiektowego rozumie się język oparty na czterech głównych zasadach [2]:

- abstrakcja danych,
- hermetyzacja (enkapsulacja),
- dziedziczność,
- polimorfizm.

Abstrakcyjny typ danych to specyfikacja klasy struktury danych. Specyfikacja opisuje strukturę danych jako listę wykonywanych przez nią czynności, a nie jako strukturę fizyczną. CA-Clipper nie posiada możliwości tworzenia abstrakcyjnych typów danych, można jedynie je symulować używając pierwotnych typów. Np. CA-Clipper nie ma możliwości odwołania się do rekordów jako całości (kilka rekordów połączonych logicznie). Dla

tych celów trzeba utworzyć oddzielny zbiór zmiennych, najczęściej tablicę, która odpowiada polom bazy danych. Na tak wyspecyfikowany formalnie zbiór danych (wykorzystując jedynie jego reprezentację) nakłada się czynności, które można realizować, np. kopiowanie, przenoszenie, kasowanie. Połączenie kodu (w tym wypadku – listy czynności) i danych (tj. tablicy) stanowi w CA-Clipper podstawę abstrakcyjnego typu danych i programowania obiektowego.

Użycie abstrakcyjnych typów danych wymaga skoncentrowania się na postaci struktury danych. Wybranie właściwego zbioru cech dla danego problemu stanowi najtrudniejsze zadanie.

Hermetyzacja to ukrywanie danych przed użytkownikiem. Dzięki hermetyzacji wewnętrzne działanie klasy nie jest widoczne dla użytkownika. Użytkownik wie, co klasa może wykonać, ale nie wie i nie musi wiedzieć, w jaki sposób to wykonuje ani jak reprezentuje dane. W związku z tym nie jest dostępny kod źródłowy standardowych klas CA-Clippera. Dokumentacja zawiera jedynie opis działania metod. Konstrukcja klasy z definicji zawiera hermetyczne typy danych. Jeśli typ danych jest hermetyczny, użytkownicy mogą uzyskać dostęp do danych tylko przez wyspecyfikowane łącze. Łącze polega na zwróceniu czegoś, dzięki czemu użytkownik może odwołać się do struktury danych, ale nie do samych danych. Przykładem jest zmienna nazwana uchwytem pliku (*Handle*) w funkcji FOpen, FRead, FSeek, FClose.

Dzięki hermetyzacji rozwiązuje się automatycznie szereg problemów. Hermetyzacja nie pozwala na zniszczenie struktury obiektu, nie można przekazać metodzie danych nieodpowiedniego typu ani przeprowadzić niedozwolonych operacji na typie danych czy też zmienić atrybutów danych. Pozwala jedynie użytkownikowi na modyfikację danych przy pomocy wywołań metod. Użytkownicy mają dostęp tylko do tych danych, które realizator klasy umieścił w łączu.

Dziedziczność umożliwia tworzenie klas, na podstawie klas już istniejących, bez potrzeby przepisania tych samych fragmentów kodu. W nowej klasie trzeba wyspecyfikować tylko różnice między nią a klasą poprzednią. Nowa klasa automatycznie dziedziczy wszystkie własności i atrybuty klasy macierzystej, dzięki czemu można wielokrotnie wykorzystać ten sam kod programu. W języku CA-Clipper 5.3 nie można jednak realizować mechanizmu dziedziczenia.

Polimorfizm polega na zdefiniowaniu takich samych nazw metod w różnych klasach. Polimorfizm jest istotny, ponieważ umożliwia pisanie metod działających w dowolnej klasie implementującej wymagane metody. W predefiniowanych klasach języka Clipper występują metody o tych samych nazwach, np. metody przenoszenia kursora *right()*, *left()*, *end()*, *home()* występują zarówno w klasie TBrowse, jak i Get.

W różnych językach programowania obiektowego lub też tylko zorientowanych obiektowo występują niewielkie różnice terminologiczne. Paradygmat OOP definiuje formalną nazwę pojedynczych elementów danych, reprezentujących klasę i nazwę operacji wykonywanych przez tę klasę. W CA-Clipperze **klasa** to specyfikacja abstrakcyjnego typu danych. Określa ona dane potrzebne do implementacji typu i operacje wykonywane na danych tego typu. W języku CA-Clipper pojedyncze elementy danych nazywają się **zmiennymi instancyjnymi**, zaś wykonywane czynności **metodami** [2]. Metody obiektu w CA-Clipperze mogą czytać i modyfikować (niektóre) wartości zmiennych instancyjnych. Biorąc pod uwagę zakres uprawnień zmiennych instancyjnych, można wyróżnić zmienne **nie eksportowane** (ukryte) i **eksportowane** [2, 4].

Zmienne nie eksportowane są ukryte wewnątrz klasy i stanowią własność obiektu. Twórca klas – firma Computer Associates International (CA) – wyspecyfikował zmienne, które nie mogą być modyfikowane przez użytkownika, gdyż mógłby on zniszczyć obiekt, zmieniając w nieodpowiedni sposób zmienną. Np. klasa TBrowse zawiera nie eksportowaną zmienną, będącą tablicą wskazań do obiektów TBColumn. Metody klasy TBrowse, takie jak *addColumn()*, *delColumn()*, *insColumn()* wykonują operacje na obiektach TBColumn, pomimo niewidoczności zmiennych i ich nazw dla programisty. Jednak każdy obiekt TBrowse musi posiadać nazwę dla zarządzania metodami TBColumn. Tworzenie zmiennych nie eksportowanych jest możliwe dzięki zasadzie hermetyzacji, ponieważ umożliwia modyfikację wewnętrznego działania klasy (przez firmę CA) bez informowania o tym użytkownika.

Zmienne eksportowane mogą być dwóch typów: przeznaczone wyłącznie do odczytu i podstawiane. Wartości zmiennych wyłącznie do odczytu są przekazane na zewnątrz jedynie jako wartości wynikowe metod. Użytkownicy klasy mogą korzystać ze zmiennych, ale nie mogą zmienić ich wartości. Zmienne podstawialne są dostępne zarówno do odczytu, jak i modyfikacji.

Dostęp do wszystkich zmiennych instancyjnych odbywa się przy pomocy operatora przesłania ":", umieszczonego między nazwą obiektu a metodą czy też zmienną. Przykładowo w klasie TBrowse odczyt i zmiana nagłówka w kolumnie pierwszej implementuje się następująco:

```
nagłowek := obTB:getColumn(1):heading                    // zmienna instancyjna heading
obTB:getColumn(1):heading := 'To jest nagłówek'          // jest do odczytu i podstawiana
```

Ilość wszystkich widzianych kolumn wyznacza się jako: *il_kolumn:=obTB:colCount*, gdzie *obTB* jest zmienną typu obiektowego, a *colCount* jest zmienną tylko do odczytu.

Nazwy i typy zmiennych instancyjnych oraz prawa dostępu do nich zostały określone przez firmę CA i ich spis wchodzi w skład dokumentacji języka [1].

Klasy predefiniowane

W języku CA-Clipper 5.x nie można samemu definiować klas. Istnieją jednak predefiniowane klasy w wersji 5.2: TBrowse, TBColumn, Error, Get i nowe dodatkowe w wersji 5.3: CheckBox, ListBox, PushButton, RadioButton, RadioGroup, MenuItem, TopBarMenu, PopUpMenu, ScrollBar.

Klasy te są wbudowane w język, nie można ich zmieniać ani rozszerzać zarówno ich metod, jak i zmiennych. Aby uzyskać elastyczność firma CA wprowadziła we wszystkich klasach zmienną instancyjną *cargo* służącą do różnych celów.
Wszystkimi ww. klasami programista posługuje się w identyczny sposób.

Obiekt to pojedynczy konkretny egzemplarz klasy. Obiekty tworzy się używając specjalnych funkcji zwanych konstruktorami. Konstruktor zwraca w wyniku obiekt. Przykładowo konstruktor klasy Error jest bezparametrowy i nazywa się *ErrorNew()*. Utworzenie nowego obiektu klasy Error odbywa się więc następująco:

```
<zmienna_obiektowa>  <instrukcja_podstawienia>  < konstruktor klasy (ewentualne parametry)>
        oE                    :=                          ErrorNew()
```

Aby odwołać się do elementów obiektowej struktury (zmiennych i metod), używa się zmiennej *oE*. Na przykład opis błędu występujący w programie można uzyskać przy pomocy:

```
opis_bladu:=oE:description
```

Przykład tworzenia i korzystania obiektów z klasy TBrowse i TBColumn

Klasa TBrowse jest programowalnym systemem przeglądania danych. Użytkownik deklaruje, co zamierza przeglądać, tworząc egzemplarz drugiej klasy TBColumn (wydruk 1). W TBColumn opisuje się wszystkie właściwości kolumny.

```
#include "inkey.ch"                  // przykład przeglądania bazy typu .dbf  o 3 polach
LOCAL kolumna
LOCAL obTB                           // zasymulowanie destruktora obiektu
LOCAL get
obTB          := TBrowseDb()         // utworzenie obiektu obTB
obTB:nTop     := 0                   // cztery zmienne definiujące okienko przeglądania
obTB:nLeft    := 0
obTB:nBottom:= MaxRow()
obTB:nRight  := MaxCol()
obTB:=addColumn( TBColumnNew(  "pierwsze pole",{ || pole1 }) ) // dodanie obiektów
obTB:=addColumn( TBColumnNew(  "drugie    pole",{ || pole2 }) ) // TBColumn do
obTB:=addColumn( TBColumnNew(  "trzecie   pole",{ || pole3 }) ) // obiektu TBrowse
obTB:configure()                     // umożliwienie aktualizacji obiektu TBrowse
WHILE .t.
      WHILE ! obTB:stabilize()       // wyświetlenie zawartości obiektu na ekranie
      ENDDO
      nKey:=Inkey(0)                 // oczekiwanie na wciśnięcie klawisza
      DO CASE                        // poruszanie się po wierszach i kolumnach
        case nKey == K_DOWN;         obTB:down()
        case nKey == K_PGDN;                obTB:pageDown()
        case nKey == K_CTRL_PGDN;           obTB:goBottom()
        case nKey == K_UP;           obTB:up()
        case nKey == K_PGUP;                obTB:pageUp()
        case nKey == K_CTRL_PGUP;           obTB:goTop()
        case nKey == K_RIGHT;        obTB:right()
        case nKey == K_LEFT;         obTB:left()
        case nKey == K_HOME;         obTB:home()
        case nKey == K_END;          obTB:end()
        case nKey == K_CTRL_LEFT;           obTB:panLeft()
        case nKey == K_CTRL_RIGHT;          obTB:panRight()
        case  nKey == K_CTRL_HOME;          obTB:panHome()
        case nKey == K_CTRL_END;     obTB:panEnd()
        case nKey == K_RETURN        // poprawianie pola
             kolumna := obTB:getColumn( obTB:colpos )
             get := GetNew( ROW(), COL(), kolumna:block,,  // obiekt GET
             kolumna:heading,, obTB:colorspec)
             READMODAL( {get} )      // uaktywnij GETa
             get := {}               // usunięcie GETa
             obTB:refreshCurrent()   // pokaż zmiany
        case nKey == K_ESC;                 exit
        case nKey == K_F2                   // kasuj kolumne 3
             kolumna:=obTB:delColumn(3)
                obTB:configure()
        case nKey == K_F3            // dodaj kolumne
             obTB:insColumn(obTB:colPos,kolumna)
                obTB:configure()
      ENDCASE
      ENDDO
```

Wydruk 1. Przykład użycia obiektów klasy TBrowse

Klasa TBrowse definiuje metody, które umożliwiają użytkownikowi dodanie kolumny obiektu TBColumn do obiektu TBrowse *(addColumn(), insColumn())* a także jego usunięcie i aktualizację *(delColumn(), setColumn())*.

Inne możliwości programowania obiektowego w CA-Clipperze

Aby uzyskać możliwość programowania obiektowego w CA-Clipperze (oprócz wykorzystania istniejących predefiniowanych klas) w aplikacjach pracujących pod kontrolą systemu operacyjnego DOS można:

1) dołączyć gotowe biblioteki obiektowe, np. *CLASS(y), Viewit class library,*
2) skorzystać z języka bardziej obiektowego, np. *C++, Pascal,*
3) skorzystać z symulacji zasad programowania obiektowego z wykorzystaniem istniejących predefiniowanych klas i preprocesora języka CA-Clipper,
4) skorzystać z nieudokumentowanych funkcji CA-Clippera tworzenia struktur.

Założeniem twórców nowych wersji języka Clipper było stworzenie bazy danych o otwartej architekturze, m.in. pozwalającej na dołączanie dodatkowych bibliotek innych firm. Obecnie na rynku istnieją wyspecjalizowane biblioteki rozszerzające możliwości programu CA-Clipper. Są to m.in. biblioteki obiektowe, czyniące język Clipper językiem obiektowym. Jest to najszybsza metoda programowania obiektowego w języku Clipper, która choć kosztowna (zakup bibliotek) szybko przynosi wymierne korzyści (upraszcza styl programowania).

W przypadku drugim należy napisać własne obiekty w wybranym języku programowania, a następnie dołączyć do aplikacji napisanej w Clipperze. Metoda ta jest pracochłonna (należy poznać sposoby tworzenia obiektów w wybranym języku), kosztowna (wymagająca zakupu innych języków programowania) i czasochłonna (następuje długi proces implementacji).

Clipper 5.3 nie jest językiem w pełni zorientowanym obiektowo (dopiero CA-Visual Objects jest takim językiem). Można zasymulować konstrukcje OOP, wykorzystując do tego preprocesor i klasy predefiniowane. Preprocesor jest ważnym elementem systemu CA-Clipper i dodatkiem do kompilatora języka Clipper. Dzięki niemu można pisać bardziej czytelny kod, nie zmniejszając przy tym jego efektywności.

Polecenia preprocesora, zwane dyrektywami, umieszcza się w programie źródłowym. W trakcie kompilacji są one przetwarzane na program wyjściowy. Program wyjściowy jest następnie analizowany i tłumaczony przez kompilator.

Do funkcji preprocesora języka Clipper należą:

- stałe symboliczne (dyrektywa: # define),
- pliki włączane (dyrektywa: # include),
- makrodefinicje kompilatora (dyrektywa: # define, stosowana do sekwencji kodu),
- kompilacja warunkowa (dyrektywa: # ifdef, # ifndef, # else, # endif, # undef),
- polecenia definiowane przez użytkownika – UDC (*User Define Command*) (dyrektywy: # command, # xcommand, # translate).

Problem symulacji programowania obiektowego w CA-Cliperze jest procesem złożonym, pracochłonnym i czasochłonnym.

Najtrudniejszą opcją wykorzystania programowania obiektowego w CA-Clipperze jest skorzystanie z nieudokumentowanych funkcji. Przez pojęcie nieudokumentowanych funkcji rozumie się te funkcje, które nie są wymienione w dokumentacji książkowej czy w innej formie, np. systemach pomocy, plikach pomocniczych itp.

Poznanie nieudokumentowanych funkcji jest możliwe przez:

- analizę modułów i bibliotek dołączonych wraz z programem;
- intuicję (czasem przypadek), analogię do innych języków;

- stosowanie dekompilatorów do programów w CA-Clipperze, np. *Valkyrie, Resque Clipper Decompiler;*
- zastosowanie translatorów języka Clipper na C/C++, np. *CodeBase;*
- z czasopism i książek omawiających problemy programowania w języku Clipper, np. *Clipper Advisor, DataBase Advisor.*
- z Internetowych list dyskusyjnych czy też spotkań w ramach klubów użytkowników CAClippera, np. *Cutty Sark Softsale Clipper Club.*

Skorzystanie z nieudokumentowanych, a więc nie gwarantowanych przez producenta funkcji może stwarzać problemy w przypadku, gdy producent oprogramowania w jakiś sposób je zmieni (zmieni nazwę, rozszerzy zakres itp.). Jest to również metoda pracochłonna, lecz przynosząca wymierne efekty w postaci zwięzłości kodu.

Możliwość tworzenia struktur przez nieudokumentowane funkcje CA-Clippera

Struktura jest to agregat składający się z dyskretnych danych. Można ją porównać do tablicy, której każdy element identyfikowany jest przez nazwę, bądź do rekordu. Zarówno rekord, jak i struktura składają się bowiem z pól. Struktury zawierają jedynie dane i jako takie są wstępem do klas zawierających dane i kod [3].

Funkcje tworzenia struktur należą do funkcji nieudokumentowanych CA-Clippera i są dostępne począwszy od wersji 5.01.
Są to ([3]):

__ClassNew(<cClassName>, <nVars>) → nHandle

Inicjuje słownik metod o nazwie <cClassName> z <nVars> polami (zmiennymi instancyjnymi). Funkcja zwraca wskazanie nHandle służące w dalszym procesie tworzenia struktury.

__ClassAdd(<nHandle>, <cVar>, <cFunc>) → Nil

Dodaje parę: zmienna instancyjna <cVar> i metoda <cFunc> do słownika metod. Jeśli nazwa zmiennej <cVar> jest poprzedzona znakiem podkreślenia, to <cFunc> jest traktowana jako funkcja przypisania wartości, jeśli nie – jako funkcja tylko do odczytu.

__ClassIns(<nHandle>) → oObject

Tworzy nową strukturę z odpowiednim słownikiem metod.

QSelf() → self

Zwraca obiekt z wnętrza metody, przy czym QSelf()[<x>] zwraca wartość pola (zmiennej instancyjnej) numer <x> obiektu (struktury).

Proces tworzenia struktury został zilustrowany na przykładzie obiektu typu wyświetlania znaku (wydruk 2). W najprostszym przypadku opis znaku powinien składać się z następujących zmiennych: dwóch współrzędnych, znaku wyświetlonego, metody pokazania (wyświetlenia) i schowania znaku.

SYMBOL 32 \f „Symbol" \s 12 \h Przykład poniższy (wydruk 2) można bardziej skomplikować dodając jeszcze kilka metod, np. koloru znaku, zamiany znaku na kod ASCII itp. Jednak w celach prezentacji pokazane metody dają wyobrażenie ze sposobu korzystania z nieudokumentowanych funkcji.

```
Scroll() ; SetPos(0,0)          // utworz obiekt
obiekt:=pn_new(0,0,"z")                 // jaki stan obiektu
QOut( obiekt:punkt() )                  // pokaz punkt
QOut( obiekt:punkt(.T.) )       // jaki stan obiektu
QOut( obiekt:punkt() )                  // jakie sa wartosci metod ?
QOut( obiekt:x )
QOut( obiekt:y )
QOut( obiekt:char )             // zmien wspolrzedne
 obiekt:x:=2
 obiekt:y:=2
 obiekt:char:="p"               // pokaz punkt
QOut( obiekt:punkt(.T.) )
inkey(0)                                // schowaj punkt
QOut( obiekt:punkt(.F.) )
RETURN

Function    pn_new( x, y, char )
 Local    ob                    // tworzona struktura
 Static   nHandle               // wskaznik do slownika metod
 IF nHandle == Nil              // pierwsze wywolanie i nalezy stworzyc slownik
   nHandle = __ClassNew( "PN", 4 )
   __ClassAdd( nHandle,  "x"   , "pn_x"   )
   __ClassAdd( nHandle, "_x"   , "pn_x"   )
   __ClassAdd( nHandle,  "y"   , "pn_y"   )
   __ClassAdd( nHandle, "_y"   , "pn_y"   )
   __ClassAdd( nHandle,  "char", "pn_char" )
   __ClassAdd( nHandle, "_char", "pn_char" )
   __ClassAdd( nHandle, "punkt", "pn_sch"  )
   __ClassAdd( nHandle,"_punkt", "pn_sch"  )
 ENDIF
 // Utworzenie nowej struktury
 ob = __ClassIns( nHandle )
 // Przypisanie zmiennym instancyjnym wartosci poczatkowych
 ob:x = x
 ob:y = y
 ob:char = char
 ob:punkt = .f.
RETURN ob

Function  pn_x( value )
 IF pcount() == 1
   RETURN ( qself()[1] := value )
 ENDIF
RETURN qself()[1]

Function  pn_y( value )
 IF pcount() == 1
   RETURN ( qself()[2] := value )
 ENDIF
RETURN qself()[2]

Function  pn_char( value )
 IF pcount() == 1
   RETURN ( qself()[3] := value )
 ENDIF
RETURN qself()[3]

Function  pn_sch( value )
 IF value == .T.
  DevPos( qself()[1], qself()[2] ) ; DevOut( qself()[3], "gr+/w+" )
 ELSEIF value == .F.
  DevPos( qself():x, qself():y ) ; DevOut( " " )
 ENDIF
 IF value == NIL
   RETURN qself()[4]
 ENDIF
RETURN ( qself()[4] := value )
```

Wydruk 2. Program wyświetlania znaku – przykład tworzenia struktur w języku CA-Clipper

Funkcje <cFund> w_ClassAdd() muszą być globalnie dostępne w całym programie, stąd nie mogą być typu Static, a ich nazwy muszą być unikalne. Natomiast nazwy zmiennych instancyjnych i metod nie muszą być unikalne – można nadawać takie same nazwy zmiennym w różnych klasach, a także stosować jako nazwy zarezerwowane słowa Clippera np. LEN czy RIGHT.

Do zmiennych instancyjnych można odwoływać się w dwojaki sposób: przez operator ":", jak i przez bezpośrednie wskazanie numeru zmiennej na liście zmiennych instancyjnych obiektu – czyli jak do elementów tablicy. Jeśli zależy nam na szybkości to należy preferować drugą metodę. Unika się wtedy przeszukiwania listy metod oraz dodatkowego wywoływania funkcji. Wadą takiego rozwiązania jest dość ścisłe przywiązanie się do konkretnej struktury zmiennych instancyjnych obiektu.

Firma CA nie zaleca wywołania żadnej funkcji zaczynającej się od znaku podkreślenia, ponieważ mogą one nie istnieć w przyszłych wersjach języka.

Korzyści wynikające z podejścia zorientowanego obiektowo

Wykorzystując podejście OOP w CA-Clipperze w stosunku do innych metod programowania uzyskuje się większą [2]:

- poprawność i elastyczność,
- odporność na błędy,
- rozszerzalność,
- możliwość wielokrotnego wykorzystania tych samych fragmentów kodu.

Programowanie zorientowane obiektowo pozwala przede wszystkim na obniżenie pracochłonności w przypadku zmian wymagań użytkownika, ponieważ zmiany dotyczą samego działania programu, a nie danych, na których się operuje. Wpływ zakresu zmian na koszt wytworzenia oprogramowania będzie znacznie mniejszy, niż w przypadku użycia innej metody programowania.

Zasady programowania obiektowego sprawiają, że niewielkie zmiany dotyczące przeznaczenia programu spowodują niewielkie, miejscowe zmiany w kodzie programu.

Program, którego elementy mogą być ponownie zastosowane, zwiększa wydajność zespołu programistów, obniżając tym samym koszty wytworzenia systemów użytkowych.

* * *

CA-Clipper bez względu na to, czy jest wyposażony w dodatkowy moduł, bibliotekę (opcja 2), symulację zasad programowania obiektowego (opcja 3) czy skorzysta się z możliwości tworzenia struktur (opcja 4), jest językiem, w którym można nauczyć się koncepcji programowania obiektowego i osiągnąć korzyści wynikające z tego podejścia. Klasy zdefiniowane w języku Clipper są na tyle wystarczające i obszerne, że umożliwiają opanowanie podstawowych konstrukcji językowych programowania obiektowego i efektywne ich użycie w praktyce.

Korzystając z ww. możliwości, a także stosując fragmenty programu w innych języku programowania obiektowego (C++, Pascal – opcja 1), można tworzyć aplikacje, które potem łatwo zostaną przeniesione na przyszłościową, w pełni obiektową wersję języka CA-Clipper.

LITERATURA

[1] Quick Reference CA-Clipper for DOS. CA, 1992
[2] Spence R.: Clipper 5.2 przewodnik programisty, t. 1. Wydawnictwo PLJ, Warszawa, 1994
[3] Strzelecki M.: Struktury w Clipperze. Zastosowania ITEM API. Seminarium „Cutty Sark Softsale Clipper Club", 20.04.1994 r, Warszawa, 1994, str. 1-3
[4] Zaręba J., Olszewski K., Gołębiowski L.: Programowanie obiektowe w języku CLIPPER 5.02 PL. Hitex Spółka z o.o., Warszawa, 1993.

---

dokończenie ze s. 7

Jedno z proponowanych rozwiązań przedstawiono na rys. Przykładowa architektura SZBD oparta jest o technikę wielobazową, w której moduł główny spełnia rolę koordynatora baz wyróżnionych ze względu na ich zróżnicowanie czasowe (i/lub przestrzenne) [3].

System składa się z czterech modułów zarządzania danymi i skojarzonych z nimi zbiorami (bazami) danych. Dzięki modułowi zarządzania danymi historycznymi (MZDH) (Historical Management Module) i historycznej bazie danych (HBD) są przechowywane i udostępniane dane z przeszłości. Dane do tej bazy są przesyłane w miarę upływu czasu, zwykle po uprzednim sprawdzeniu ich poprawności. Ze względu na ich niezmienność powinny być przechowywane na nośnikach tylko do odczytu, np. dyski optyczne.

Lokalny SZBD wraz z bazą danych (BBD) jest odpowiedzialny za manipulowanie „stanem bieżącym" bazy. Baza taka jest konwencjonalną bazą z konwencjonalnym SZBD, w którym czas może być wprowadzany jako atrybut. Z tej bazy wszystkie dane historyczne po ich odpowiednim zakwalifikowaniu zostają przesłane do HBD.

Odpowiedzialnym za kwalifikację, spójność i poprawność danych przesyłanych z BBD do HBD jest globalny moduł zarządzania (GMZ – Global Management Module). Pomocnicza baza danych (DBD – Auxiliary Data Base) wraz z towarzyszącym jej modułem zarządzania służy do przechowywania danych nie kwalifikujących się do żadnej innej bazy, ale koniecznych do udzielenia odpowiedzi na niektóre pytania. Przykładem takich danych mogą być hipotezy, alternatywne plany, scenariusze. Dane o przyszłych, ustalonych stanach lub o stanach przewidywanych niealternatywnie są przechowywane w bazie danych przyszłych (BDP – Future Data Base). W odpowiednim czasie dane z tej bazy są przesyłane do BBD.

Zapytania kierowane są zawsze do GMZ, który dokonuje oceny poprawności pytania, dekomponuje je, optymalizuje i przesyła odpowiednio skonstruowane pytania cząstkowe do podsystemów. Każdy podległy moduł odzyskuje odpowiednie dane (także z wielu baz) i przesyła do modułu głównego konstruującego końcowe rozwiązanie.

Budowa systemu zarządzania informacją o charakterze temporalnym napotyka także inne trudności związane, np. z ogromną ilością danych zawartych w bazach. Fizyczna reprezentacja takich danych jest kolejnym, bardzo ważnym aspektem budowy SZBD, jednak problemy jej dotyczące wykraczają poza zakres niniejszego artykułu.

Kolejny artykuł jest poświęcony omówieniu temporalnych baz danych oraz konstrukcjom językowym pozwalającym na odzyskiwanie z nich informacji.

LITERATURA

[1] Alagar V.S.: Modeling Time in Information Systems. w: Formal Methods in Database and Software Engineering. Alagar V.S., Lakshaman L., Sadri F. (Eds.) Springer-Verlag, 1993
[2] Ben-Zvi J.:The Time Relational Model. Rozprawa doktorska, University of California, 1982
[3] Ling D., Bell D.: Modelling and Managing Time in Database Systems. The Computer Journal, vol. 35, no. 4, 1992
[4] Ling D., Bell D.: Taxonomy of Time Models in Databases., Information and Software Technology, vol. 32, no. 8, 1990
[5] Lundberg B.: An Axiomatization of Events., BIT, no. 22, 1982
[6] Roddick J.F. Patrick J.D.:Temporal Semantics in Information Systems – A Survey. Information Systems, vol. 17, no. 3, 1992
[7] Snodgrass R.: Temporal Databases. Materiały konferencji Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, Piza, 1992
[8] Theodoulidis C., Loucopoulos P.:The time dimension in conceptual modelling. Information Systems, vol 16, no. 3, 1991.

GANDALF stawia na rozwój sieci dostępowych i ISDN

Stanisław Michalak

Gandalf Polska Ltd. jest kanadyjską firmą „high-tech", specjalizującą się w projektowaniu, produkcji i instalacji urządzeń oraz systemów telekomunikacyjnych i teleinformatycznych. Gandalf jest spółką akcyjną, niezależną pod względem finansowym, organizacyjnym i technologicznym od firm komputerowych oraz od operatorów sieci. Umożliwia to neutralność i budowę optymalnych rozwiązań, integrujących różne systemy komputerowe i wykorzystujących infrastrukturę sieci publicznych od stosunkowo prostych aż do najnowocześniejszych, opartych na technologii światłowodowej, satelitarnej i radioliniowej.

Gandalf dysponuje bezpośrednim doświadczeniem we wdrażaniu m.in. następujących technologii:

- *ISDN*, szczególnie pod kątem teleinformatycznym, głównie od strony użytkownika (*CPE*). Tandemy z telefonicznymi centralami abonenckimi umożliwiające integrację głosu i danych.
- Technologie pakietowe, od *X.25*, poprzez *frame relay* aż do *cell relay (ATM)*.
- Technologie *TDM* – szybkie multipleksery pracujące w trybie podziału pasma i umożliwiające budowę systemów hierarchicznych zarówno po stronie operatora sieci, jak i użytkownika końcowego.
- Technologie *LAN*: systemy dystrybucji i okablowania strukturalnego, światłowody (wliczając *FDDI*), *bridges, routers, gateways*, zarządzanie siecią itp.
- Urządzenia transmisyjne; modemy, *data over voice*, pasmo podstawowe, modemy na zasięgi metropolitalne itp.

Gandalf Polska Ltd. specjalizuje się w nowoczesnej technologii sieciowej i oferuje szeroki zakres rozwiązań opartych głównie na sprzęcie oraz oprogramowaniu projektowanym i produkowanym przez międzynarodową grupę firm Gandalf, kierowanych przez kanadyjską firmę macierzystą Gandalf Technologies Inc. oraz na wybranych technologiach innych wiodących producentów.

Gandalf był w Polsce pionierem rozwiązań *LAN* opartych na okablowaniu strukturalnym oraz integracji sieci lokalnych i rozległych.

W chwili obecnej Gandalf instaluje w Polsce najnowocześniejsze technologie, takie jak:

- sieci *LAN* oparte na okablowaniu strukturalnym;
- sieci światłowodowe o dużej przepustowości z integracją telefonii i danych;
- sieci oparte na technologii szybkiego pakietu;
- systemy *ISDN* z integracją telefonii i danych;
- wielofunkcyjne sieci LAN/WAN z użyciem technologii *router, gateway* oraz *bridge;*
- szybka komunikacja pomiędzy sieciami LAN z użyciem urządzeń na *podczerwień;*
- bardzo szybkie połączenia sieciowe LAN-LAN, dzięki opatentowanej metodzie kompresji wraz z elementami zabezpieczeń przed nieuprawnionym dostępem.

Rozwiązania firmy Gandalf są wykorzystywane w chwili obecnej w wielu kluczowych instytucjach w Polsce: bankowości, szkolnictwie wyższym, administracji państwowej, energetyce, handlu detalicznym i firmach zagranicznych działających w kraju.

Po tak krótkim przedstawieniu firmy omówimy poniżej kierunki rozwoju sieci dostępowych i dziedziny, w których firma Gandalf może zaoferować swoje produkty i usługi.

Tendencje rozwoju sieci dostępowych

Rozwój systemów telekomunikacyjnych w ostatnich latach przebiega bardzo burzliwie. Wprowadzenie techniki światłowodowej do sieci podstawowej (bazowej) stało się faktem w wielu krajach. Jednakże wprowadzenie światłowodów do sieci abonenckiej nie jest już tak oczywiste, wymaga uwzględnienia wielu czynników i doboru specyficznych rozwiązań dla konkretnych warunków terenowych i lokalizacji obiektów. W praktyce telekomunikacyjnej istnieje wyraźny podział na sieci międzycentralowe i sieci abonenckie. Sieci międzycentralowe buduje się już od dłuższego czasu, zarówno z zastosowaniem techniki światłowodowej, jak i z wykorzystaniem urządzeń transmisyjnych pracujących w technologii PDH, a obecnie wprowadza się technologię SDH. Natomiast w sieciach abonenckich wykorzystuje się jeszcze w zdecydowanej większości technikę kabli miedzianych. Dopiero pełna cyfryzacja central telefonicznych i możliwość tworzenia cyfrowych kanałów transmisji danych stwarza możliwości rozwoju nowych usług telekomunikacyjnych i teleinformatycznych.

Zgodnie z tendencjami światowymi, prawdopodobnie już za kilka lat większość nowo budowanych instalacji abonenckich będzie instalacjami zawierającymi światłowody – sieci FITL (Fibre in the Loop). Wprowadzenie bardziej efektywnych sieci dostępowych ma na celu stworzenie nowej, wielofunkcyjnej infrastruktury telekomunikacyjnej dla zapewnienia w przyszłości integracji usług i możliwości realizacji wszystkich podstawowych form komunikacji: telefonii, transmisji telewizyjnych tekstów i obrazów, szybkiej transmisji danych komputerowych, zwłaszcza pomiędzy sieciami LAN (po łączach kablowych, radiowych i satelitarnych). Dopiero duża szybkość przesyłania w łączach abonenckich pozwoli na rozwój usług multimedialnych i realizację wzrastającego zapotrzebowania na sieci szerokopasmowe. Jak opisują niektóre opracowania [2], w najbliższych latach rozwój sieci telekomunikacyjnych powinien podążać w kierunku:

- integracji technik transmisji i komutacji,
- integracji usług telekomunikacyjnych,
- powszechnego stosowania światłowodów jako medium transmisji,
- scentralizowania systemu nadzoru i zarządzania siecią,
- wprowadzenia technik asynchronicznego trybu transferu informacji ATM.

Powyższe tendencje będą miały duży wpływ na rozwój abonenckich sieci dostępowych.

Techniki budowy sieci dostępowych

Budowa sieci dostępowych ma na celu stworzenie możliwości korzystania przez użytkowników za pośrednictwem urządzeń końcowych z resursów przesyłowych sieci szkieletowych oraz z dostępu do baz informacyjnych i usług telematycznych. We współczesnych sieciach rozległych stosuje się różnorodne technologie i protokoły transmisyjne poczynając od: X.25, TDM, ISDN a kończąc na: Frame Relay i ATM. Wszystkie te technologie zdążają do stworzenia elastycznej wielousługowej sieci, która będzie w stanie zaspokoić wszystkie oczekiwania klientów.

Jakość transmisji w sieciach dostępowych w dużym stopniu zależy od wyboru medium transmisyjnego (kabla). Ponieważ od wielu lat parametry techniczno-ekonomiczne światłowodów są lepsze od innych rodzajów kabli, zatem właściwy wybór medium transmisyjnego dotyczy w zasadzie wyboru rodzaju łącza światłowodowego.

Światłowody produkuje się w trzech zasadniczych odmianach. Są to: **światłowody skokowe, gradientowe** i **jednomodowe.** Poza konstrukcją, różnią się one od siebie przede wszystkim użyteczną szerokością pasma i sprawnością optyczną sprzęgania ze źródłem promieniowania.

Światłowody skokowe mają najniższą szerokość pasma, rzędu 20 MHz/km oraz najwyższą sprawność sprzęgania ze źródłem światła, około 70 %. Ponieważ w światłowodach skokowych dominują zniekształcenia modowe, są one odpowiednim medium transmisyjnym do współpracy z diodami elektroluminescencyjnymi, których szerokie widmo promieniowania i tak powoduje wystąpienie znacznej dyspersji materiałowej [1].

Światłowody jednomodowe wykazują najwyższą szerokość pasma, często powyżej 100 GHz/km, natomiast sprawność sprzęgania w ich przypadku spada do około 20%. Pełne wykorzystanie ich możliwości transmisyjnych wymaga jednak zastosowania jako źródła światła diody laserowej lub nawet diody laserowej jednomodowej. Ceny światłowodów jednomodowych są około 3 razy niższe niż kwarcowych światłowodów wielomodomych.

Światłowody gradientowe zajmują pozycję pośrednią z szerokością pasma około 1 GHz/km i sprawnością sprzęgania około 40 %. Mogą one efektywnie współpracować z diodami laserowymi lub z diodami elektroluminescecyjnymi, ale tylko w zakresie drugiego okna widmowego. Powyższe dane dotyczą światłowodów ze szkła kwarcowego.

Stosuje się również tanie **światłowody plastermerowe**, jednak wysoka tłumienność ogranicza ich zastosowanie do bardzo krótkich odcinków sieci.

Należy podkreślić, że pełne wykorzystanie wszystkich zalet światłowodów wymaga przestrzegania opracowanych norm środowiskowych dotyczących m.in. zakresu temperatur, wibracji, poziomu skażenia środowiska i zapylenia, które są nieco inne od stosowanych w przypadku sieci miedzianych, i muszą być bezwzględnie przestrzegane.

W systemach światłowodowych zwielokrotnianie sygnałów można realizować na wiele sposobów. Najczęściej spotykane metody to:

- zwielokrotnianie przestrzenne (SDM – Space Division Multiplexing) polegające na powieleniu toru transmisyjnego, łącznie z urządzeniami wejścia i wyjścia,
- zwielokrotnianie kierunkowe (DDM – Direction Division Multiplexing) polegające na rozdzieleniu ruchu w jedną i drugą stronę,
- zwielokrotnianie falowe (WDM – Wavelength Division Multiplexing) polegające na równoległym zastosowaniu wielu źródeł generujących różne długości fali oraz odpowiadającej im liczbie fotodetektorów,
- zwielokrotnianie czasowe (TDM – Time Division Multiplexing) polegające na przeplataniu okien czasowych poszczególnym przyporządkowanym kanałom,
- zwielokrotnianie kodowe (CDM – Code Division Multiplexing) polega na niezależnym kodowaniu i dekodowaniu sygnału każdego z kanałów oraz na przesyłaniu zakodowanych sygnałów łącznym torem szerokopasmowym,
- zwielokrotnianie częstotliwościowe (FDM – Frequency Division Multiplexing) można zrealizować stosując oddzielne źródła podczerwieni o nieco innej długości fali dla każdego kanału.

Przyszłościowe techniki przełączania optycznego najprawdopodobniej będą opierały się na systemie ATM (Asynchronous Transfer Mode), polegającym na przesyłaniu informacji w komórkach zawierających zwykle 48 bajtów, poprzedzonych 5-cioma bajtami adresowymi. W razie potrzeby 53 bajtowe komórki ATM mogą być bez trudu przesyłane w sieciach synchronicznych standardu SDH lub pleziochronicznych PDH. System ATM pozwala wykorzystać wszystkie zalety zwielokrotniania czasowego TDM, w praktyce oznacza to możliwość jednoczesnej transmisji sygnałów wideo, danych komputerowych i sygnałów mowy. Wielką zaletą ATM-u jest łatwość realizacji systemów przełączających, odporność na kolizje, zatłoczenie i blokowanie sieci. System ATM jest postrzegany jako niezastąpiony przy przechodzeniu do szerokopasmowej abonenckiej sieci cyfrowej.

Struktury istniejących abonenckich sieci światłowodowych ODN (Optical Distribution Network) są bardzo różnorodne. Cztery podstawowe konfiguracje to:

- gwiazda,
- podwójna gwiazda,
- pętla,
- szyna z odgałęzieniami.

Sieci nie zawierające elementów wzmacniających określa się dodatkowo terminem PON (Passive Optical Network).

Konfiguracja gwiazdy (star) zapewnia bezpośrednie połączenie abonenta z centralą, co jest bardzo korzystne przy szerokopasmowych usługach interakcyjnych, jednak – w przypadku sieci pokrywających duży obszar – może wymagać stosowania światłowodów jednomodowych i wysokiej klasy źródeł generacji światła.

Konfiguracja podwójnej lub nawet wielokrotnej gwiazdy umożliwia zastosowanie tańszych źródeł i światło-

wodów wielomodowych, przynajmniej w krańcowych rejonach sieci.

Konfiguracja pętli (ring) jest zalecana do wzajemnego łączenia dużej liczby równoważnych abonentów. Przy zastosowaniu podwójnej pętli, transmisja może odbywać się w dwóch niezależnych kierunkach, co pozwala na zwiększenie elastyczności i niezawodności działania sieci. Konfiguracja szyny z odgałęzieniami jest najtańszą spośród wyżej wymienionych; może być przydatna w linearnie zorientowanych osiedlach mieszkaniowych.

Szczegółowa topologia abonenckiej sieci światłowodowej zależy od głębokości wnikania w pobliże abonenta. Analizę oferowanych systemów należy rozpocząć od wprowadzenia jednolitego systemu oznaczeń. Można tu wyróżnić szereg standardów takich jak:

- pętla abonencka, realizowana z wykorzystaniem światłowodu zamiast pary miedzianej FITL (Fiber In The Loop);
- łącze do mieszkania FTTH (Fibre To The Home), architektura systemu FITL z optycznymi zakończeniami umieszczonymi w mieszkaniu, przy czym jedno zakończenie obsługuje jednego abonenta;
- łącze do biura FTTO (Fiber To The Office), architektura systemu FITL najczęściej punkt-punkt, dająca szeroko pasmowy styk optyczny;
- łącze do budynku FTTB (Fiber To The Building), architektura systemu FITL z optycznymi zakończeniami sieci umieszczonymi w budynku i obsługującymi grupę abonentów;
- łącze do krawężnika FTTC (Fiber To The Curb), optyczne zakończenia sieci obsługujące więcej niż jednego abonenta, umieszczone są w szafkach ulicznych;
- łącze do piętra FTTF (Fiber To The Floor);
- łącze do strefy FTTZ (Fiber To The Zone);
- łącze do biurka FTTD (Fiber To The Desk).

Wybór konkretnego rozwiązania zależy głównie od przesłanek ekonomicznych. Za budową optycznych sieci dostępowych przemawiają ich podstawowe zalety:

- praktycznie nieograniczone pasmo przepustowe,
- możliwość realizacji usług tradycyjnych, obecnie wdrażanych usług telematycznych w sieciach komputerowych i zintegrowanych usług w sieciach ISDN,
- możliwość łatwego podłączania do sieci SDH,
- możliwość scentralizowanego zarządzania sieciami.

Oczekiwania odbiorców nowych technologii

Nowoczesne sieci dostępowe powinny umożliwiać transmisję różnorodnych sygnałów. Wymagają tego następujące powody:

- abonenci mają coraz większe wymagania związane nie tylko z połączeniami telefonicznymi; wymagany jest dostęp do sieci transmisji danych cyfrowych, banków danych, zrodziły się potrzeby transmisji mulitimedialnych, itp;
- przy dzisiejszych cenach kabli i urządzeń, dla niektórych konfiguracji obiektów, korzystniejsze cenowo są oferty urządzeń wykorzystujących technikę światłowodową;
- światłowodowe sieci dostępu umożliwiają lepszą jakość transmisji, łatwość utrzymania w sprawności sieci;
- połączenie światłowodowych sieci dostępu z siecią transmisji TV znacznie zmniejsza sumaryczne koszty instalacji i potencjalnie umożliwia doprowadzenie tych sygnałów, nawet do dość rozproszonych i odległych abonentów.

Generalnie stopień zaawansowania technicznego sieci miejscowych zależy od wielkości ruchu i zaopatrzenia na nowe usługi. Na ogół operatorzy telekomunikacyjni wprowadzają nowe technologie „wyspowo" w miejscach, gdzie jest to opłacalne. Pierwszymi klientami wyposażanymi w sieci światłowodowe są centra finansowe i bankowe, które wymagają szybkiej wymiany informacji i różnorodnych usług.

Podstawowe oczekiwania klietów sprowadzają się do:

- dostępu do szerokiego wachlarza usług świadczonych przez sieci,
- zintegrowanego sposobu świadczonych usług; klient nie chce budować osobnych sieci dla pracy w różnych systemach, oczekuje jednego kanału dostępowego do różnych sieci,
- korzystania z bogatej oferty programów i informacji,
- prostoty obsługi, najlepiej prawie intuicyjnej,
- doskonałej jakości oglądanych obrazów i słuchanego dźwięku,
- możliwości selektywnego wyłączania dowolnego urządzenia z systemu,
- bezpieczeństwa przesyłanej informacji,

a najważniejsze – to niska cena wszelkich usług.

Marzenia i dążenia klientów do działania w czasie rzeczywistym zawsze przewyższały możliwości systemów telekomunikacyjnych.

Kompleksowość obsługi klienta

Firma Gandalf, jeden z głównych dostawców inteligentnych urządzeń do budowy sieci lokalnych, miejskich i rozległych, ujawniła swoim klientom strategię integracji usług i kierunki wspierania rozwoju usług ISDN w Polsce, w szczególności usług tzw. wideokonferencji.

Firma gwarantuje niezawodne urządzenia, fachową pomoc i szkolenie, doskonały serwis. Jakość i solidność firmy wbudowana jest w naturalny sposób w struktury techniczne i systemowe produktów.

W rzeczywistości oznacza to dostarczanie różnorodnych kombinacji urządzeń do:

- realizacji dostępu po łączu komutowanym do Internetu,
- budowy lokalnych sieci komputerowych,
- realizacji transmisji danych po łączach komutowanych (analogowych i cyfrowych),
- realizacji dostępu do sieci ISDN,
- realizacji transmisji po łączach radiowych i na podczerwień,
- tworzenia systemu okablowania strukturalnego dowolnie wskazanych obiektów,
- wspólnego wykorzystywania pasma agregatowego,
- agregacji wolnych kanałów dostępowych w jeden agregatowy.

Firma Gandalf bierze na siebie pełną odpowiedzialność za zbudowanie i uruchomienie dowolnej podsieci abonenckiej, spełniającej wymagania klienta.

Dotychczas firma Gandalf głównie podejmowała się:

- wykonywania okablowania strukturalnego budynków pod przyszłe sieci komputerowe,
- oferowania sprzętu i oprogramowania dla integracji sprzętu różnych producentów, dla transmisji danych.

Aktualnie poszerzerzono zakres działań firmy o:

- wykonywanie ekspertyz i ocen możliwości budowy

sieci teleinformatycznych dla klientów tzw. dużego i średniego businesu,
- opracowywanie projektów i budowę zintegrowanych systemów wymiany informacji, umożliwiających jednoczesną transmisję danych, głosu i obrazu,
- badanie i testowanie jakości łączy zarówno analogowych, jak i cyfrowych.

Zajęto się również wdrażaniem nowych technologii w zakresie transmisji danych, m.in.:
- projektowaniem i budową sieci ISDN,
- pilotowymi wdrożeniami wideokonferencji,
- budową sieci komputerowych z protokołami komunikacyjnymi typu: FDDI, Frame Relay, ATM,
- realizacją transmisji danych z szybkością 2Mbps po łączach naturalnych – miedzianych,
- dostarczaniem nowoczesnych urządzeń typu: bridge, router, brouter, multiplekser statystyczny.

Podstawowe zasady postępowania firmy przy obsłudze klientów to propozycje optymalnych dla nich rozwiązań uwzględniających takie czynniki jak: minimalizacja kosztów budowanych systemów, otwartość i elastyczność rozwiązań oraz ewolucyjność projektów poprzez dopasowanie ich do istniejącej infrastruktury telekomunikacyjnej.

Powyżej wymienieniono jedynie przykładowe tematy, którymi zajmuje się firma Gandalf, zgodnie z oczekiwaniami klientów.

Integracja usług domeną firmy Gandalf

Wśród ekspertów od prognoz rozwoju telekomunikacji panuje pogląd, że przyszła sieć telekomunikacyjna to sieć światłowodowa z bogatą paletą usług, ukierunkowana na obsługę tzw. klientów biznesowych, nie zapominając o klientach prywatnych. Trzeba jednak brać pod uwagę fakt, że obecnie większość sieci abonenckich zbudowana jest w oparciu o miedziane kable symetryczne. Aktualne możliwości to stosowanie sieci mieszanych: światłowody w części międzycentralowej, a kable miedziane w części abonenckiej, jednak z tendencją wprowadzania światłowodu głęboko w sieć dystrybucyjną oraz eliminacją skrętki miedzianej – linii o zbyt małej przepustowości. Dla zwiększenia pojemności istniejących sieci abonenckich stosuje się systemy PCM-N i HDSL, zaś do budowy nowych części magistralnych sieci dostępu stosuje się światłowody zakończone koncentratorami. To rozwiązanie ułatwia doprowadzenie światłowodów w miarę napływu zgłoszeń od pojedynczych abonentów reflektujących na usługi szerokopasmowe.

Odrębną, ale niezmiernie istotną rolę w modernizacji i rozbudowie dostępu mają radiowe systemy abonenckie i systemy transmisji na podczerwień. Ich wyjątkowo pożyteczną cechą jest możliwość praktycznie natychmiastowego przyłączenia nowego abonenta o dowolnej lokalizacji na obszarze zasięgu danego systemu. Możliwości tych systemów to między innymi:
- szybka realizacja sieci przyłączy stacjonarnych, przenośnych, mobilnych,
- szybka realizacja sieci rozdzielczych,
- szybka realizacja sieci magistralnych,
- szybka realizacja sieci o zasięgu lokalnym.

W tym obszarze sieci abonenckich firma Gandalf dysponuje szeregiem doskonałych urządzeń, przydatnych do ich budowy.

Wraz z rozwojem sieci dostępowych zmienia się również zakres świadczonych usług przez firmę Gandalf, od zwykłego przekazu telefonicznego, poprzez transmisję danych za pomocą modemów i za pośrednictwem sieci komputerowych do usług zintegrowanych i szerokopasmowych w sieciach ISDN i B-ISDN.

Gandalf prekursorem widekonferencji w Polsce

Tradycyjna sieć telefoniczna, powstała z myślą o analogowym przesyłaniu głosu, aktualnie nie spełnia współczesnych wymagań dotyczących transmisji danych. Trwające od wielu lat prace nad rozwojem sieci telekomunikacyjnych łączących możliwości sieci telefonicznych i sieci transmisji danych zakończyły się powstaniem sieci ISDN, umożliwiających cyfrową transmisję danych i sygnalizację, integrację różnego typu usług i standaryzację dostępu. Transmisja danych w ISDN, niezależnie od ich rodzaju (głos, dane komputerowe, obraz), jest realizowana w oparciu o technologię cyfrową, włączając w to urządzenia abonenckie typu: aparat telefoniczny, fax, komputer. Komunikacja w sieci ISDN odbywa się cyfrowo, bez względu na rodzaj usługi. Różne formy informacji dostarczanej przez użytkownika są kodowane cyfrowo przy wejściu do sieci ISDN i przesyłane w niej za pomocą kanałów transmisyjnych:
- kanału B o przepustowości 64 kbps,
- kanału D o przepustowości 16 kbps.

Kanał B (bearer channel) służy do transmisji danych użytkownika, natomiast kanał D (control channel) używany jest głównie do sygnalizacji.

Przyłącze abonenckie może być realizowane u abonenta w następujących konfiguracjach:
- Basic Rate User Access składający się z 2 kanałów B o prędkości 64 kbps każdy oraz jednego kanału D o prędkości 16 kbps. Jest to podstawowa konfiguracja portu ISDN, a sieć często nazywa się siecią wąskopasmową.
- Primary Rate User Access oferujący 30 kanałów B o prędkości 64 kbps każdy oraz jednego kanału D o prędkości 64 kbps. Jest to konfiguracja szerokopasmowa (30B + D), oznaczana czasami B-ISDN. Do jednego portu ISDN można dołączyć jednocześnie 8 urządzeń końcowych.

Ze względu na dużą szybkość i jakość transmisji sieci ISDN pozwalają na realizację szeregu atrakcyjnych usług dla użytkowników. Między innymi możliwa jest:
- realizacja połączeń telefonicznych o wysokiej jakości kanałów rozmównych i realizacja szeregu dodatkowych udogodnień typu: informacja o numerze abonenta dzwoniącego, itp.;
- realizacja szybkiej, bezbłędnej i taniej transmisji danych komputerowych;
- budowa sieci lokalnych i rozległych sieci komputerowych.

Jeszcze większe możliwości stwarzają sieci B-ISDN, które pozwalają na realizację usług takich jak:
- wideotelefonia i wideokonferencje,
- superszybka transmisja danych (przesyłanie dużych zbiorów między komputerami),
- szybkie telesterowanie,
- przesyłanie obrazów o dużej rozdzielczości,
- telefax kolorowy,
- poczta wizyjna (redagowanie z jednoczesnym drukowaniem),

- telezakupy i telezlecenia,
- teleedukacja, i inne.

Sprzęt firmy Gandalf stanowi doskonałą platformę do wdrażania usług ISDN. Dzięki modularnej budowie może w sposób elastyczny obsługiwać zarówno abonentów analogowych, jak i abonentów cyfrowych. Firma Gandalf zapoczątkowała wdrażanie technologii ISDN na terenie Polski już w latach osiemdziesiątych i zajmuje obecnie czołową pozycję w stosowaniu zaawansowanych usług ISDN w szczególności wideokonferencji, o czym mogło się przekonać szerokie grono uczestników ostatnio odbytych w Warszawie europejskich warsztatów ISDN pod tytułem Workshop ISDN'96. Zaprezentowany sprzęt i łatwość realizacji dzięki niemu wielu wyrafinowanych usług telekomunikacyjnych świadczą o wysokiej marce produktów firmy Gandalf. Najwyższej jakości usługi wideokonferencyjne są realizowane na bazie sprzętu produkowanego przez firmę Picturetel (56% rynku światowego), z którą firma Gandalf współpracuje od 1989 roku.

Stworzone na potrzeby cyfrowych sieci zintegrowanych usługowo rozwiązania techniczne firmy Gandalf mogą być wykorzystane przy uzyskiwaniu dostępu do sieci Internet. Zamiast modemu do komputera wystarczy podłączyć kartę ISDN i można korzystać ze zwykłych linii telefonicznych, ale uzyskiwana prędkość jest pięć razy większa. Chociaż dostęp do Internetu przez łącze ISDN jest droższy, to jednak jego popularność rośnie i w przyszłości będzie jedną z dominujących form wymiany informacji. Tam, gdzie nie jest możliwe skorzystanie z usług sieci ISDN, firma Gandalf proponuje superszybki dostęp przez urządzenia o nazwie „Etherphone" produkcji firmy PainGain.

System zarządzania sieciami

Urządzenia firmy Gandalf oraz inne urządzenia zgodne ze standardem SNMP MIB II objęte są systemem zarządzania siecią Gandalf *Passport*.

System zarządzania siecią Gandalf *Passport* jest rozwiązaniem unikalnym, gdyż pozwala na pełną integrację elementów LAN posiadających funkcję *SNMP MIB*, jak również sieci rozległych pracujących w trybie podziału pasma *(TDM), ISDN, cell switching* wraz z obsługą funkcji wielomedialnych.

Passport realizuje następujące funkcje w zintegrowanych sieciach wielomedialnych:

- wszystkie aspekty sieci przedstawione są na jednej konsoli graficznej,
- ułatwione jest dodawanie nowych aplikacji wraz z ujednoliconym dostępem przez interfejs użytkownika *MOTIF*,
- wewnętrzna baza danych umożliwia analizę statystyk sieciowych, edycję raportów i wymianę informacji z zewnętrznymi bazami danych,
- obsługa urządzeń i systemów innych producentów odbywa się na podstawie zgodności protokołów SNMP,
- ułatwione jest dodawanie nowych *MIBów*,
- monitorowanie sieci i rekonfiguracja odbywa się za pośrednictwem konsoli z natychmiastowym wyświetleniem alarmów o różnych priorytetach,
- użytkownik określa zindywidualizowane priorytety i selekcję typów alarmów wraz ze sposobem ich eskalacji za pośrednictwem funkcji *Object Manager*,
- zbieranie danych o wydajności i przepustowości sieci ułatwia strojenie sieci oraz optymalizację wykorzystania jej zasobów,
- automatyczna funkcja *discovery* zapewnia automatyczne włączenie w system zarządzania nowo podłączonych urządzeń,
- szybkie wykrywanie powtarzających się adresów w sieci lub nieuprawnionych (nielegalnych) urządzeń zwiększa ogólne bezpieczeństwo funkcjonowania sieci.

System *Passport* jest dostępny w wersji dla mniejszych użytkowników (do 256 węzłów), zrealizowany w oparciu o stację roboczą PC/MS-Windows oraz w wersji SUN Sparkstation o praktycznie nieograniczonej liczbie węzłów.

* * *

Wspólną zaletą przedstawionych powyżej rozwiązań jest zastosowanie jednolitej, sprawdzonej, wciąż rozbudowywanej i udoskonalanej linii produktów sieciowych firmy Gandalf. Zastosowanie produktów firmy Gandalf pozwala na stopniową rozbudowę sieci i zakup kolejnych elementów w miarę wzrostu ilości użytkowników sieci i wzrostu wymagań względem sieci (zarządzalność, łączenie LAN-WAN).

LITERATURA

[1] Kowalski A., Rezulski M.: Metody budowy światłowodowych sieci dostępu, materiały szkoleniowe IŁ 1995r

[2] Zagrobelny T.: Tendencje rozwoju sieci dostępu w świecie, materiały szkoleniowe IŁ 1995 r.

O/21/96

Zaprosili nas :

Polska Izba Informatyki i Telekomunikacji na Śniadanie Klubowe Izby, podczas którego został przedstawiony Manifest Telekomunikacyjny i problem opodatkowania sprzętu komputerowego z tytułu prawa autorskiego

Microsoft na konferencję prasową poświęconą problemom piractwa komputerowego w Polsce i na świecie

INFORMIX na Koktail z okazji podpisania Umowy Ramowej o Współpracy pomiędzy Urzędem Rady Ministrów i Informix Software sp. z o.o.

IBM Polska oraz **TCH COMPONENTS** na konferencję prasową, informującą o podpisaniu umowy dystrybutorskiej

OPTIMUS SA na prezentację najnowszych przedsięwzięć spółki i holdingu OPTIMUS SA w zakresie integracji systemów, technologii sieciowej oraz produkcji najnowszych wieloprocesorowych serwerów z Pentium Pro ®

Wiesław Migut na wieczór sympatyków TELEINFO

Centrum Promocji Informatyki na forum „ŚWIAT BAZ DANYCH"

ERICSSON na konferencję prasową, podczas której zaprezentowane zostały nowe modele cyfrowych telefonów komórkowych GSM

ComPress S.A. na spotkanie z przedstawicielami firmy IBM poświęcone implementacji IBM Software Strategy w Polsce oraz planom IBM dotyczącym polskiego rynku

Intergraph Europe (Polska) na pokaz nowych stacji TDZ z nową grafiką New RealiZm

IDG Poland S.A. do odwiedzenia II Międzynarodowej Wystawy Technik Telekomunikacyjnych i Sieciowych ComNet Warszawa 96. **(ap)**

SYSTEMY CZASU RZECZYWISTEGO

REDAGUJE ZESPÓŁ: dr inż. Zbigniew FRYŹLEWICZ (Politechnika Wrocławska), prof. dr hab. inż. Janusz GÓRSKI (Francusko-Polska Wyższa Szkoła Nowych Technik Informatyczno-Komunikacyjnych w Poznaniu), prof. dr hab. inż. Zbigniew HUZAR (Politechnika Wrocławska), dr inż. Krzysztof SACHA (Politechnika Warszawska), prof. dr hab. inż. Tomasz SZMUC (Akademia Górniczo-Hutnicza), dr inż. Kazimierz WORWA (Wojskowa Akademia Techniczna), mgr inż. Zdzisław ŻURAKOWSKI (Instytut Automatyki Systemów Energetycznych we Wrocławiu) – przewodniczący zespołu.

ROK III ● SIERPIEŃ ● 1996 ● 21

Hierarchiczne tworzenie aplikacji czasu rzeczywistego

Tomasz Szmuc
Piotr Szwed
Katedra Automatyki Akademii Górniczo-Hutniczej

Specyfikacja, projektowanie i implementacja oprogramowania czasu rzeczywistego do sterowania procesów jest zagadnieniem złożonym, wymagającym wiedzy interdyscyplinarnej. Tradycyjnie jest to obszar działania automatyków (identyfikacja, modelowanie, synteza algorytmów sterowania) oraz informatyków (wybór środowiska – system operacyjny, języki programowania, specyfikacja, projektowanie i weryfikacja oprogramowania czasu rzeczywistego).

Konstrukcja metod i pakietów wspomagających wymaga zazwyczaj znajomości obu dziedzin. Złożoność procesu tworzenia oprogramowania do sterowania jest uwarunkowana nie tylko strukturą implementowanych algorytmów sterowania, lecz również dodatkowymi cechami jak: współbieżność, wymagania dotyczące czasu odpowiedzi systemu oraz niezawodności. Weryfikacja poprawności ma szerszy zakres i jest znacznie istotniejsza niż w przypadku „klasycznych systemów".

W ostatnim okresie zaczęły się pojawiać zaawansowane otoczenia programowe dla systemów CACSD [4]. Przykładowo systemy ORCCAD (Open Robot Controller Computer Aided Design) [7], COTS i BEACON [3] to specjalizowane otoczenia, które są rozwijane w ośrodkach uniwersyteckich i czołowych firmach (GEC). Z drugiej strony, komercyjnie dostępne są zaawansowane systemy rozwijania programów sterujących, na przykład SIMULINK Real-Time Workshop firmy Mathworks, Inc. oraz SystemBuild/AutoCode firmy Integrated Systems Inc. Oferują one łatwe w użyciu techniki projektowania systemów sterowania typu *rapid prototyping*. Wspomaganie dotyczy jednak projektowania systemów wykorzystujących raczej klasyczne algorytmy sterowania i to jedynie na poziomie funkcji realizowanej przez algorytm. Brakuje narzędzi wspomagających pozostałe, niezwykle ważne i trudne fazy (etapy) implementacji algorytmów sterowania. Należą do nich: określenie struktury oprogramowania (podział na moduły, warstwy), synchronizacja i komunikacja między komponentami, weryfikacja poprawności na różnych poziomach (w tym również całego) systemu. Dostępne systemy nie są więc przystosowane do projektowania złożonych, wielopoziomowych, rozproszonych systemów.

Obserwuje się także rozwój oprogramowania wspomagającego projektowanie systemów sterowania implementowanych w środowisku systemów operacyjnych czasu rzeczywistego. Przykładem takiego rozwiązania jest ControlShell firmy Real-Time Innovation [4] bazujący na systemie operacyjnym VxWorks.

Znaczna komplikacja procesu tworzenia aplikacji czasu rzeczywistego, jak również dostępność różnych narzędzi (środowisk) wspomagających, skłoniły do próby utworzenia środowiska zintegrowanego, wspomagającego w możliwie szerokim zakresie tworzenie aplikacji czasu rzeczywistego. Utworzenie tego typu środowiska wymaga nie tylko integracji poszczególnych podsystemów, lecz również implementacji funkcji, które nie są dostarczane przez wspomniane wyżej systemy wspomagające.

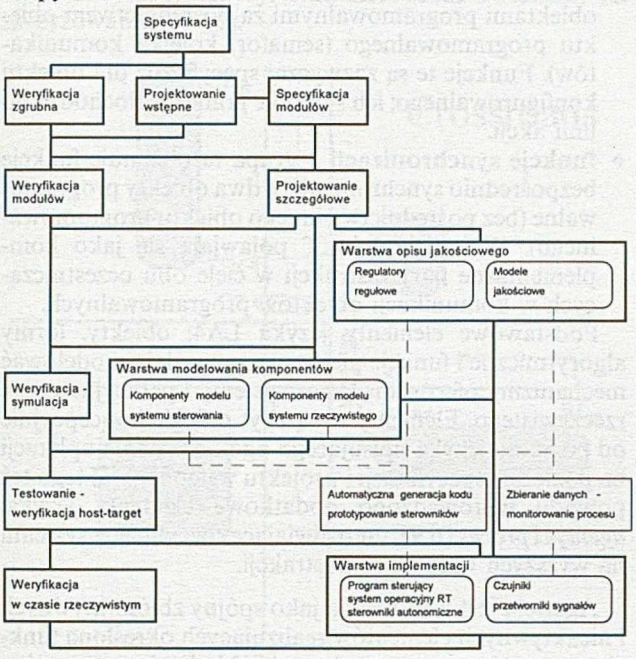
W pracy można wyróżnić dwie główne części. Pierwsza z nich zawiera opis koncepcji systemu zintegrowanego do rozwijania aplikacji czasu rzeczywistego, natomiast druga pokazuje tworzenie tej aplikacji w fazach specyfikacji i projektowania.

Koncepcja zintegrowanego systemu do tworzenia aplikacji czasu rzeczywistego

Cykl życia oprogramowania charakteryzuje w ostatnich latach ciągła modyfikacja. Wynika to z rozwoju narzędzi

wspomagających poszczególne fazy cyklu i – co za tym idzie – z możliwości łączenia faz lub generacji nowych ścieżek. Odwołując się zatem do modelu bazowego [14] oraz uwzględniając propozycje modyfikacji [1,2] proponuje się przebieg faz tworzenia oprogramowania pokazany na rys. 1. Szczególną uwagę należy zwrócić na następujące zagadnienia:

- Hierarchiczne tworzenie aplikacji z weryfikacją poprawności na poszczególnych etapach i poziomach,
- Zastosowanie pakietów wspomagających kolejne etapy.



Rys. 1. Środowisko do tworzenia oprogramowania czasu rzeczywistego

Rozbicie procesu tworzenia na poszczególne warstwy ma ułatwić korzystanie z poszczególnych pakietów wspomagających, pozwala również przeprowadzać weryfikacje poprawności w sposób bardziej systematyczny, z wykorzystaniem informacji dostępnej na danym poziomie.

Dążenie do efektywnego wykorzystania istniejących pakietów wspomagających odnosi się głównie do sfery dziedzinowej projektu (tworzenie systemu sterowania). Wynika to zazwyczaj z dość trudnego zadania syntezy algorytmu sterowania, jak również z możliwości wykorzystania zaawansowanych środowisk wspomagających. Nie oznacza to, że istniejące pakiety wspomagające obsługują cały obszar tworzenia aplikacji czasu rzeczywistego. Brakuje zwłaszcza efektywnych narzędzi wspomagających warstwę opisu jakościowego. Podobną ocenę można zastosować do etapów weryfikacji poprawności poszczególnych faz (prawa strona rys.1). Stąd istotna jest konstrukcja odpowiednich narzędzi wspomagających, jak również integracja wszystkich elementów (modułów).

Zapronowano, aby środowiskiem integrującym pozostałe pakiety był system VxWorks. Główną zaletą systemu VxWorks jest skrośna architektura host-target, pozwalająca na „interakcyjną analizę" wykonania aplikacji [10].

Umożliwia to rozszerzenie analizy poprawności przez symulację na komputerze host, daje narzędzia wspomagające testowanie/analizę poprawności w układzie host-target, jak również ułatwia analizę aplikacji w czasie rzeczywistym. Wymienione własności oraz możliwości współpracy z różnymi pakietami zadecydowały o wyborze tego systemu.

Przedstawiony poniżej krótki opis poszczególnych faz ma na celu wyjaśnienie przyjętej koncepcji proponowanego środowiska.

Specyfikacja będzie tworzona z wykorzystaniem informacji z bazy danych. Na tej podstawie będzie określona specyfikacja tworzonego oprogramowania. Faza projektowania wstępnego dotyczyć będzie dekompozycji, funkcji, parametrów i interfejsów, jak również częściowo struktur danych i mechanizmów komunikacji i synchronizacji. Weryfikacja zgrubna będzie odnosić się do analizy poprawności zaproponowanej struktury i po udowodnieniu tej poprawności (w tym ogólnym sensie) umożliwi generację ram dla poszczególnych modułów [11]. Kolejne fazy dotyczą projektowania szczegółowego (wypełnianie ram określonych w poprzedniej fazie). Przewiduje się korzystanie z pakietów wspomagających (Matlab, SIMULINK), jak również opracowanie i implementację odpowiednich systemów wspomagających projektowanie w dziedzinie nieklasycznych algorytmów sterowania (procesy dyskretne, systemy z bazą wiedzy). Projektowanie będzie połączone z generacją kodu, automatycznie (jeśli możliwe), w trybie interakcyjnym (łączenie parametryzowanych modułów bibliotecznych), ewentualnie przez ręczne programowanie. Ta ostatnia możliwość będzie eliminowana przez opracowanie odpowiednich bibliotek. Weryfikacja poprawności poszczególnych modułów jest realizowana w kolejnej fazie. Po pomyślnym zakończeniu tej fazy otrzymuje się kolekcję modułów poprawnych z punktu widzenia struktury i powiązań (weryfikacja zgrubna) oraz pod względem realizowanych funkcji (weryfikacja modułów). Kolejny etap dotyczy integracji oprogramowania czasu rzeczywistego ze środowiskiem, w którym ma ono pracować (system operacyjny). Przy tworzeniu aplikacji przewiduje się zastosowanie techniki skrośnego tworzenia aplikacji (cross development). Stąd faza integracji będzie początkowo przebiegać na komputerze głównym (host) i na tym komputerze będzie realizowana symulacja-weryfikacja takiego systemu. W następnym etapie oprogramowanie sterujące będzie przeniesione (przez sieć) na komputer docelowy (target), pracujący pod kontrolą systemu operacyjnego czasu rzeczywistego [10].

Specyfikacja, projektowanie i weryfikacja systemów czasu rzeczywistego

W niniejszym rozdziale pokazane będzie zastosowanie języka LA4 do specyfikacji i projektowania (szczególnie fazy wstępnej) aplikacji czasu rzeczywistego. Opisane będą również zagadnienia weryfikacji poprawności odpowiadające tym etapom.

Charakterystyka języka LA4

Język LA4 [5,6,12,13] jest zdefiniowany w dwóch postaciach: graficznej (LA4_G) i tekstowej (LA4_T). Specyfikacja LA4_G jest dwuwymiarowym diagramem opisującym strukturę wielozadaniowej aplikacji czasu rzeczywistego. Schemat zawiera graficzną reprezentację wszystkich statycznych składowych aplikacji (*obiektów*) oraz mechanizmów synchronizacji i komunikacji (realizowanych

przez wywołania *funkcji pierwotnych*). Obiekt przedstawiany jest na schemacie w postaci symbolu graficznego uzupełnionego listą parametrów. Wywołania funkcji są zaznaczane jako połączenia między obiektami (zwykle linie łamane z dołączonym symbolem graficznym, pozwalającym rozróżnić typ funkcji). Powyższe elementy są uzupełnione przez formy algorytmiczne odpowiadające strukturalnym instrukcjom języków programowania wysokiego poziomu. Gramatyka języka LA4 definuje ściśle symbolikę używaną do reprezentacji obiektów i funkcji, a także reguły połączeń.

Kod LA4_T jest równoważny syntaktycznie diagramowi LA4_G i jest zazwyczaj generowany na podstawie LA4_G.

Budowa aplikacji za pomocą systemu LA4 składa się z dwóch etapów. W pierwszym projektuje się aplikację używając w tym celu narzędzia graficznego; otrzymany schemat wzajemnych powiązań obiektów (LA4_G) jest automatycznie tłumaczony do równoważnej postaci kodu (LA4_T). Następnie w zależności od wybranego docelowego systemu operacyjnego i języka programowania jest uruchamiany odpowiedni moduł translatora generującego szkielet programu czasu rzeczywistego.

Podobieństwo zasadniczych elementów i mechanizmów jądra systemów operacyjnych czasu rzeczywistego pozwoliło jednak na wyodrębnienie nadzbioru obejmującego najczęściej używane typy obiektów i funkcji i zdefiniowanie go w gramatyce języka LA4.

Obiekty LA4. Rozróżnia się dwa typy obiektów: *programowalne* i *konfigurowalne*. Do pierwszej grupy zalicza się: zadanie (*task*), procedurę obsługi przerwania (*interrupt*) i procedurę uśpioną (*alarm*). Obiekty programowalne są odzwierciedlone w kodzie LA4_T jako procedury zawierające ciągi instrukcji odpowiadające wołanym funkcjom pierwotnym. Obiekty konfigurowalne: semafor (*semaphore*), kolejka komunikatów (*mailbox*), komunikat (*message*), tablica komunikatów (*message pool*), zasób (*resource*) i zdarzenie (*event*), jawnie nie wykonują żadnych działań, dlatego też w kodzie LA4_T pojawiają się jedynie jako argumenty wywołań funkcji pierwotnych.

Dla obiektów języka LA4_G definiuje się (w zależności od typu) linie stanu (*state bars*), linie wywołań (*progress bars*) i ewentualnie linie akcji (*action bars*). *Linia stanu* jest zdefiniowana dla każdego obiektu jako docelowy region dla linii łamanych reprezentujących funkcje pierwotne kreacji, destrukcji i zmiany stanu. *Linia wywołań* jest zdefiniowana jedynie dla obiektów programowalnych. Jest to region, z którego odchodzą linie reprezentujące funkcje pierwotne wywoływane przez obiekt programowalny. Kolejność punktów zaczepienia na linii wywołań jest zgodna z sekwencją wykonywanych instrukcji. Linie wywołań obiektów programowalnych mogą zostać rozszerzone i rozgałęzione przez użycie form algorytmicznych *IF...THEN...ELSE, WHILE, REPET...UNTIL, SWITCH...CASE, FOREVER* (pętla nieskończona) oraz *PROCEDURE*. Symbole form algorytmicznych określają alternatywne drogi przepływu sterowania. Linie akcji występują w obiektach konfigurowalnych. W tych regionach umieszczane są końce linii łamanych funkcji pierwotnych specyficznych dla danego obiektu programowalnego.

Funkcje pierwotne LA4. Funkcje pierwotne LA4 zapewniają jednolity, niezależny od systemu operacyjnego interfejs do funkcji systemowych. Odpowiednikiem funkcji LA4 jest wywołanie jednej lub kilku funkcji jądra systemu operacyjnego. Wyróżnia się trzy zasadnicze grupy funkcji pierwotnych:

- **funkcje stanu** – pozwalają na zmianę lub odczyt stanu obiektu. Należą do nich globalne funkcje kreacji i destrukcji lub funkcje typu aktywacji, uśpienia zadań, maskowania przerwań. W gramatyce LA4_G docelowymi regionami dla graficznych symboli funkcji stanu są zawyczaj linie stanu obiektów.
- **funkcje akcji** – umożliwiają ustanowienie połączenia (synchronizacji lub komunikacji) pomiędzy dwoma obiektami programowalnymi za pośrednictwem obiektu programowalnego (semafor, kolejka komunikatów). Funkcje te są zazwyczaj specyficzne dla obiektu konfigurowalnego; ich symbole graficzne dochodzą do linii akcji.
- **funkcje synchronizacji** – grupa ta obejmuje funkcje bezpośrednio synchronizujące dwa obiekty programowalne (bez pośrednictwa innego obiektu programowalnego). W kodzie LA4_T pojawiają się jako komplementarne pary instrukcji w ciele obu uczestniczących w komunikacji obiektów programowalnych.

Podstawowe elementy języka LA4: obiekty, formy algorytmiczne i funkcje pierwotne pozwalają modelować mechanizmy oferowane przez systemy operacyjne czasu rzeczywistego. Elementy te są zbyt odległe koncepcyjnie od poziomu języka opisującego ogólne zadania aplikacji na poziomie specyfikacji i projektu wstępnego. Z tego też powodu wprowadzono dodatkowe elementy języka: *agencję* i *proces* [6,9], umożliwiające specyfikację systemu na wyższych warstwach abstrakcji.

Agencja jest traktowana jako spójny zbiór aktywnych i nieaktywnych elementów realizujących określoną funkcję wewnątrz programu. Agencja składa się z procesów, czyli aktywnych elementów nie mogących istnieć w pełni samodzielnie oraz kanałów komunikacyjnych, których rolę pełnią klasyczne obiekty konfigurowalne jak semafor, kolejka komunikatów czy zasób określonego typu. Komunikacja między procesami odbywa się za pośrednictwem kanałów komunikacyjnych z użyciem standardowych funkcji pierwotnych, jak *SEND_TO_SEM, WAIT_ON_MBX*. Definiując agencję wyróżnia się także jej elementy terminalne (*borders*). Mogą one reprezentować obiekty LA4 pełniące rolę interfejsu do obiektów fizycznych (czujnik, terminal), kanały komunikacji przy współpracy z innymi agencjami aplikacji lub też punkty startowe procedur inicjalizacji i destrukcji.

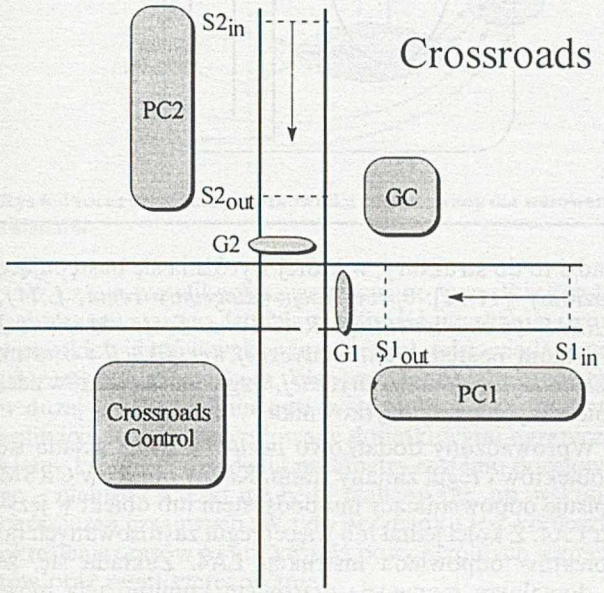
Proces reprezentuje aktywną funkcję składową aplikacji. Może on być hierarchicznie rozłożony na podprocesy. Z punktu widzenia projektowania najważniejsze jest wydzielenie jego interfejsu, tzn. podanie zbioru niezbędnych kanałów komunikacyjnych i obsługujących je funkcji pierwotnych. Sama implementacja procesu może być zrealizowana na najniższym szczeblu hierarchii z użyciem języka LA4 lub z wykorzystaniem języka programowania docelowej platformy.

Cykl projektowania aplikacji polega więc w fazie wstępnej na podziale jej funkcji między agencje i podprocesy, doborze obiektów pełniących rolę kanałów komunikacji, czyli na wydzieleniu jej samodzielnych elementów składowych i ustaleniu interfejsu komunikacji i synchronizacji. W dalszej fazie uszczegółowiania projektu jest przeprowadzana dekompozycja procesów na podprocesy lub modelowanie bezpośrednio z użyciem standardowych elementów języka.

Powyższy opis dotyczył charekterystycznych cech języka LA4, przydatnych z punktu widzenia specyfikacji i projektowania aplikacji czasu rzeczywistego. Dokładniejsze informacje dotyczące języka można znaleźć w [13].

Specyfikacja i projektowanie aplikacji – przykład

Rozważmy przykład systemu sterowania światłami na skrzyżowaniu (rys. 2).



Rys. 2. Skrzyżowanie – rysunek poglądowy

Sygnalizacja świetlna (*G1* i *G2*) jest bezpośrednio sterowana przez moduł *Gate Control (GC)*. Na każdej ulicy zainstalowano po dwa czujniki wykrywające nadjeżdżające samochody. Czujnik S_{in}, wykrywa pojazdy wjeżdżające do *Passage Zone*, natomiast drugi S_{out}, jest umieszczony bezpośrednio przed skrzyżowaniem. Czujniki te generują przerwania, które są obsługiwane przez moduł *Passage Control (PC)*. System jest sterowany przez moduł *Crossroads Control (CC)*.

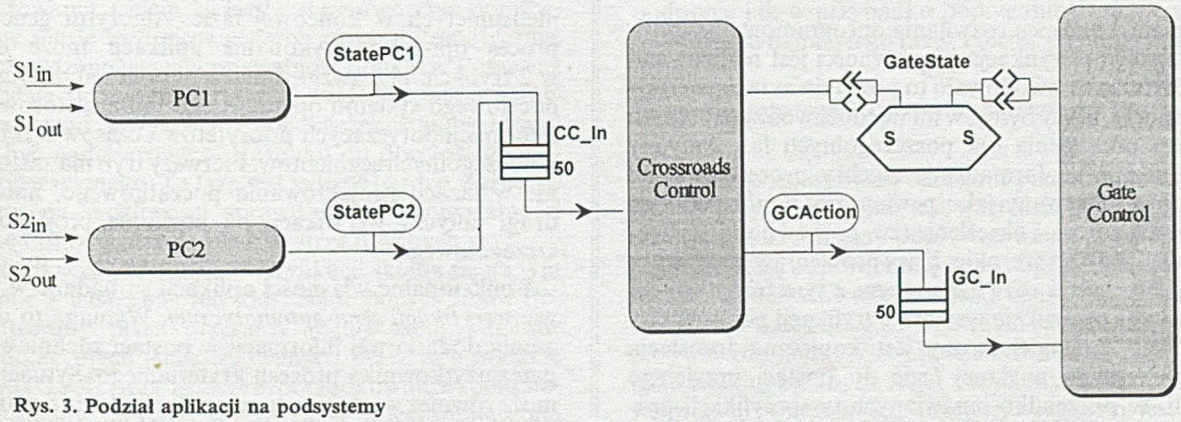
Powiązania między poszczególnymi modułami na najwyższym poziomie hierarchii pokazano na rys. 3. Cztery podsystemy komunikują się za pośrednictwem trzech kanałów komunikacyjnych: *CC_IN* (wejście do skrzynki pośredniczącej w wymianie informacji z modułem *Crossroads Control*), *GC_In* (wejście dla modułu *Gate Control*) oraz *Gate State* – dzielona pamięć z ochroną za pomocą semafora.

Dwa identyczne podsystemy *PC1* i *PC2* obsługują przerwania generowane przez czujniki i modyfikują wektor stanu w części dotyczącej ilości pojazdów w *Passage Zone* i aktualnie maksymalnego czasu oczekiwania pojazdów w tej strefie – t_{max}. Przykład specyfikacji takiego podsystemu (*PC*) podano w pracy [12]. Modyfikacja parametru t_{max} może być realizowana przez zmianę, rejestrację i porównywanie momentów czasowych pojawiania się przerwań od czujników. Informacja o zmianie stanu (*StatePC*) jest wysyłana do skrzynki *CC_In*. W ten sposób jest możliwa również sygnalizacja innych sytuacji, np. wystąpienie błędu, przekroczenie maksymalnego czasu oczekiwania.

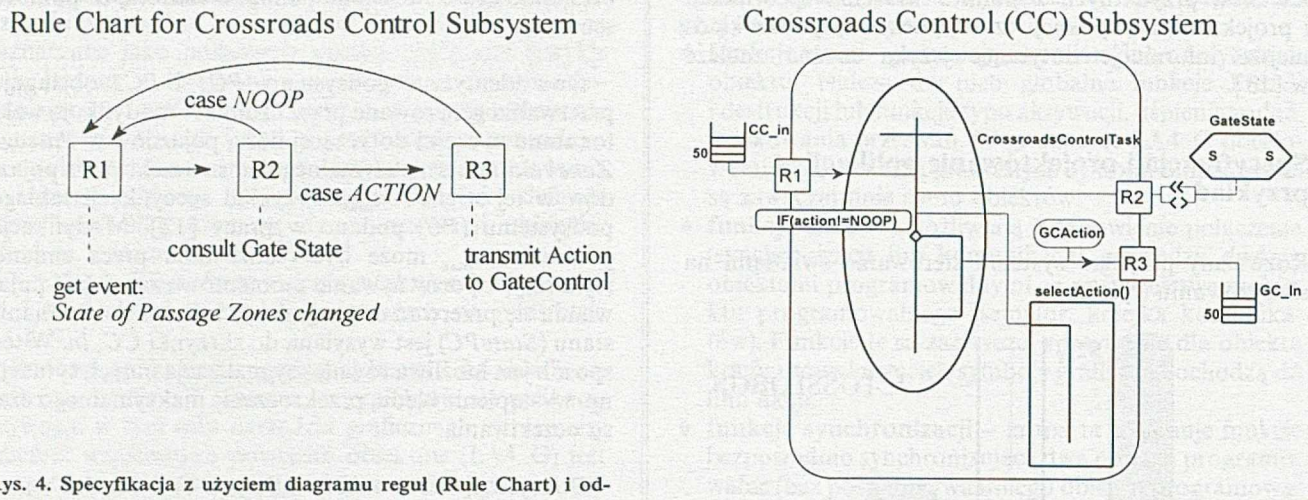
Podsystem sterowania światłami – *Crossroads Control (CC)* – pełni rolę głównej pętli decyzyjnej dla tworzonej aplikacji. Podsystem ten jest w stanie oczekiwania na zdarzenia generowane przez moduły *PC1* i *PC2*. Przesłanie wiadomości oznacza zmianę stanu w kontrolowanym obszarze (np. pojazd wjechał lub wyjechał z tego obszaru). Wówczas jest sprawdzany aktualny stan bramki (stan zasobu *GateState* modyfikowany przez podsystem *GC*) i następnie podejmowana jest decyzja względem stanu bramki (zmiany świateł). Decyzja ta jest przekazywana do podsystemu *Gate Control* za pośrednictwem skrzynki *GC_In*.

Specyfikacja tego podsystemu z wykorzystaniem diagramu reguł (Rule Chart) i odpowiadająca jej implementacja są przedstawione na rys. 4. Wewnątrz tego podsystemu zaznaczono procedurę *selectAction()*, implementującą algorytm sterowania światłami na skrzyżowaniu. Algorytm ten, skonstruowany jest na bazie inżynierii ruchu).

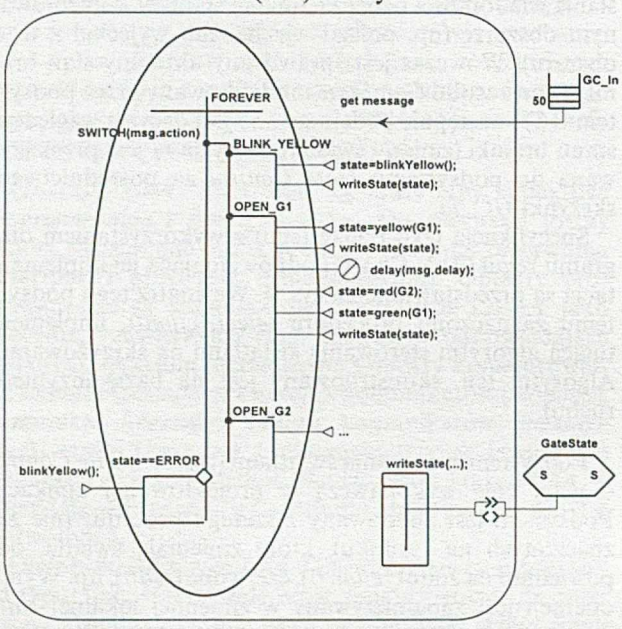
Podsystem sterowania światłami (rys. 5) – *Gate Control* – pełni rolę wykonawczą w projektowanej aplikacji. Podsystem jest zbudowany z szeregu procedur (nie zaznaczonych na rysunku), które zmieniają światła, odpowiednio na żółte (*yellow()*), czerwone (*red()*), itp. Wynik operacji jest zapamiętywany w zmiennej lokalnej *state* i następnie wpisywany do zmiennej dzielonej *GateState*. Pętla główna składa się z instrukcji oczekiwania na wiadomość ze skrzynki *GC_In* i następnie na podstawie żądanej akcji (np. *Open_G1*) wykonuje sekwencję wywołań odpowiednich funkcji (*yellow(), delay(), red(G2), green(G1)*).



Rys. 3. Podział aplikacji na podsystemy



Rys. 4. Specyfikacja z użyciem diagramu reguł (Rule Chart) i odpowiadająca jej implementacja z użyciem języka LA4



Rys. 5. Specyfikacja podsystemu sterowania światłami z użyciem LA4

Wielowarstwowa weryfikacja specyfikacji

W przyjętej koncepcji rozwijania oprogramowania czasu rzeczywistego weryfikacja poprawności jest realizowana w każdej fazie. Spowodowało to poszukiwania uniwersalnego modelu, który byłby, w miarę możliwości, wspólnym punktem odniesienia dla poszczególnych faz. Przyjęty model bazuje na formalizmie algebry procesów, gdzie proces jest opisywany jako pewien graf. Ogólność tego pojęcia, jak również określenie tzw. poprawności względnej umożliwia opis szerokiej klasy problemów poprawności [8]. Narzędzia rozwijane razem z tym formalizmem umożliwiły konstrukcję systemu weryfikacji poprawności względnej. Z drugiej strony jest konieczna translacja z opisu systemu w danej fazie do postaci przyjętego modelu. W przypadku omawianych tu specyfikacji prowadzi to do struktury, w której wyróżnia się następujące poziomy [11,12]: *specyfikacji (diagramy regul, LA4), CRSM (Communicating Real-Time State Machines)* – poziom pośredni, *automatycznej weryfikacji* i ostatni: *weryfikacji semi-automatycznej*, względem kryteriów zdefiniowanych przez użytkownika.

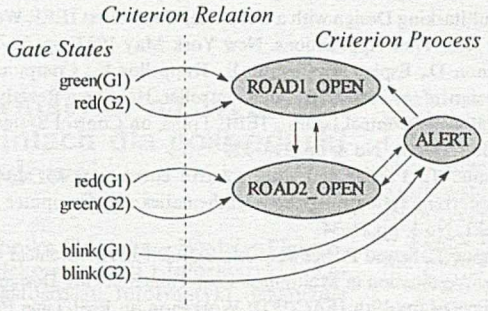
Wprowadzony dodatkowo *poziom CRMS* składa się z obiektów i reguł zmiany stanu. Każdy obiekt w CRSM opisuje odpowiadający mu podsystem lub obiekt w języku LA4. Z kolei jedna lub więcej reguł zastosowanych do obiektów odpowiada instrukcji LA4. Zakłada się, że w dowolnym momencie czasowym komunikacja może wystąpić między co najwyżej dwoma obiektami. Każda reguła ma postać dozoru i instrukcji/ciągu instrukcji, zmieniających stan obiektów. Nie wszystkie akcje są dozwolone w danym stanie i tego typu ograniczenia, związane z językiem są dołączone jako predefiniowane dla danego obiektu procesy kryterialne. Pozostałe wymagania, odnoszące się do własności funkcjonalnych, są określane przez procesy kryterialne, zdefiniowane przez użytkownika. Specyfikują one sekwencje stanów uważane za poprawne.

W *warstwie automatycznej* weryfikacji poprawności są realizowane dwie podstawowe funkcje: generowany jest proces opisujący wykonanie aplikacji (na podstawie opisu CRSM) oraz jest wykonywana weryfikacja poprawności względem predefiniowanych procesów kryterialnych. Dotyczy to tzw. własności wykonywalności, np. wystąpienie błędów, zakleszczenie, pojawienie się obiektów nieusuniętych w końcowej fazie. Algorytm generujący proces opisujący wykonanie aplikacji może działać w 2 trybach: bez uwzględnienia ograniczeń dotyczących docelowego systemu operacyjnego lub z zastosowaniem informacji dotyczących priorytetów i czasów wykonania poszczególnych segmentów. Pierwszy tryb ma zastosowanie w fazach projektowania początkowego, natomiast drugi dotyczy weryfikacji na poziomie projektowania szczegółowego.

Funkcjonalne własności aplikacji są badane w warstwie *weryfikacji semi-automatycznej*. Wymaga to dostarczenia dodatkowej informacji w postaci zdefiniowanego przez użytkownika procesu kryterialnego. Sytuacja taka może również wystąpić w przypadku badania ograniczeń,

które również można sprowadzić do badania częściowej lub całkowitej poprawności.

Przykład kryterium poprawności zdefiniowanego przez użytkownika pokazano na rys. 6. Zauważmy, że sytuacja, gdzie mamy do czynienia z dwiema otwartymi drogami (*green(G1), green(G2)*) będzie uznana za niepoprawną, jeśli zdefiniujemy poprawność jako całkowitą [8].



Rys.6. Proces kryterialny użytkownika zdefiniowany dla sterowania światłami

W czasie weryfikacji są generowane wszystkie ścieżki (sekwencje) stanów. Istnieje możliwość uzyskania zbioru wszystkich ciągów wykonanych reguł, odpowiadających poszczególnym ścieżkom. Wspomniane zbiory mogą być w dużej mierze nadmiarowe w stosunku do rzeczywistej aplikacji. Jest to spowodowane dodatkowymi ograniczeniami, które wprowadzają parametry systemu docelowego – niektóre ścieżki nie będą realizowane, np. wskutek ograniczeń czasowych. W tym przypadku jest konieczne określenie czasów wykonywania poszczególnych segmentów oraz reguł szeregowania.

Hierarchiczna weryfikacja

Hierarchiczna weryfikacja poprawności jest realizowana w etapach specyfikacji i projektowania, odpowiadających kolejnym poziomom w hierarchii podsystemów. Polega to na transformacji zbioru wymagań dotyczących aplikacji na zbiór wymagań dotyczących struktury i zbioru wymagań dotyczących funkcji podsystemów. Weryfikacja jest realizowana w trzech etapach: *weryfikacja zgrubna, weryfikacja podsystemów, weryfikacja zależności czasowych.*

Weryfikacja zgrubna

Weryfikacja zgrubna jest wykonywana we wstępnej fazie projektowania. Celem tej fazy jest sprawdzenie poprawności w sensie kryteriów generowanych automatycznie, tzn. sprawdzana jest wykonywalność aplikacji. Ponieważ nie podaje się żadnych założeń względem parametrów systemu, stąd wynikowy zbiór ścieżek stanów (reguł) może być nadmiarowy w stosunku do docelowego systemu operacyjnego. Ta faza jest jednak bardzo ważna, ze względu na możliwości sprawdzenia wykonywalności aplikacji oraz wykrycia błędów strukturalnych.

Specyfikacja podlegająca weryfikacji składa się na tym poziomie z następujących klas: zbiór podsystemów (reprezentowany przez diagramy reguł), zbiór kanałów komunikacyjnych (specyfikowanych przez odpowiednie elementy LA4) oraz lista przyporządkowań wspomnianych kanałów do reguł. Specyfikacja ta jest tłumaczona na odpowiadającą jej postać, zapisaną w języku CRSM i następnie przekazana do systemu weryfikacji omówionego wyżej. W czasie translacji dokonuje się sprawdzenia poprawności (syntaktycznej) przypisań kanałów do reguł. Standardowy zbiór obiektów języka LA4 jest rozszerzany o obiekt modelujący podsystem. Obiekt ten zawiera następujące elementy: diagram reguł, informacja o fazie wykonania (*INITIAL, KERNEL, FINAL*) oraz stan, tj. odsyłacz do ostatnio wykonywanej reguły. Aktualny stan pozwala określić zbiór reguł, które mają być wykonane jako następne, względem porządku określonego przez diagram.

Proces kryterialny jest generowany dla tej fazy automatycznie i umożliwia sprawdzenie wystąpienia zakleszczenia, blokowania, poprawności kolejności faz *INITIAL, KERNEL, FINAL* itp.

Weryfikacja podsystemów

W tej fazie jest realizowana weryfikacja podsystemów, specyfikowanych przy użyciu języka LA4. Głównym jej celem jest sprawdzenie odpowiedniości (poprawności względnej) pomiędzy tymi poziomami opisu. Specyfikacja wyższego poziomu (diagram reguł) tworzy proces kryterialny, opis z użyciem języka LA4 jest procesem weryfikowanym. Weryfikacja obejmuje dwa typy kryteriów: generowanie automatyczne (wykonywalność) i przekaz z poziomu wyższego (własności funkcjonalne).

W wyniku weryfikacji otrzymuje się następujące rezultaty:

- sprawdzenie zgodności specyfikacji LA4 z wymaganiami określonymi przez diagram reguł,
- wygenerowanie czasowego diagramu, jeśli są dołączone wartości czasów poszczególnych przejść.

Rozważa się dwa rodzaje charakterystyk czasowych:

A. Dokładne. Poszczególne segmenty kodu są wartościowane przez czas, tworząc tym samym drzewo czasów wykonania tych segmentów. Poszczególne segmenty są klasyfikowane jako *przerywalne(I), nieprzerywalne (N)* oraz oczekujące na *zdarzenie(E)*. Drzewo jest budowane począwszy od najniższego poziomu specyfikacji, a następnie przechodzi się przez poszczególne poziomy integracji systemu.

B. Zgrubne. W tym przypadku czas przejścia jest określony przez trójkę $\tau = (TA, TD, TE)$, gdzie TA oznacza czas potrzebny na wykonanie przejścia, TD reprezentuje czas opóźnienia, natomiast TE czas oczekiwania na zdarzenie. Dla pewnych przejść czasy te są określane w postaci przedziałów [minimalny, maksymalny]. Budowanie takiego drzewa czasów wykonania odbywa się podobnie jak w przypadku poprzednim, tzn. począwszy od poziomu najniższego przez kolejne poziomy integracji.

Weryfikacja ograniczeń czasowych

W tej fazie wymagania są określone przez proces kryterialny, w którym każdy łuk jest wartościowany przez przedział czasu określający poprawne momenty (przedziały) pojawienia się odpowiednich reguł. Specyfikacja aplikacji zawiera dodatkowe informacje, tj. charakterystyki czasowe, jak również regułę szeregowania zadań (priorytetowa z wywłaszczaniem). Weryfikacja odbywa się przez symulację według ogólnych reguł określonych dla algorytmu weryfikacji poprawności. Celem tej weryfikacji jest spraw-

dzenie spełnienia zależności czasowych przy założeniu wykonywania aplikacji na docelowym komputerze [9].

* * *

Niniejsze opracowanie należy traktować jako uzupełnienie poprzednich [8-12], opisujących wybrane narzędzia/komponenty do tworzenia aplikacji czasu rzeczywistego. Prezentowany system pozwala zintegrować te narzędzia ułatwiając wykonywanie wielu iteracji w obrębie wybranych ścieżek i zwiekszając tym samym stopień weryfikacji tworzonej aplikacji i możliwości zastosowania poszczególnych narzędzi. W drugiej części pokazano zastosowanie rozwijanych wcześniej narzędzi (LA4 i poprawność względna) do specyfikacji i projektowania aplikacji czasu rzeczywistego. Przedstawiono podejście hierarchiczne, zarówno w dziedzinie tworzenia, jak i weryfikacji poprawności. Zaproponowane środowisko, jak również jego składniki są obecnie rozwijanie w Katedrze Automatyki Akademii Górniczo-Hutniczej.

Praca została wykonana w ramach grantu KBN8T11A01308: *Metody formalne i narzędzia wspomagające projektowanie i analizę oprogramowania czasu rzeczywistego do sterowania procesami.*

LITERATURA

[1] Braek R., Haugen O.: Engineering Real Time Systems. Prentice Hall
[2] Hinchey M.G.: Visual Methods in Real-Time Programming. Proceedings of the 19th IFAC/IFIP Workshop on Real-Time Programming 1994, Reichenau, 22–24 June 1994, Pergamon Press
[3] Rimvall C.M., Spang III H.A. et.al.: An Open Architecture for Automatic Code Generation using the BEACON CACE Environment. IEEE/IFAC Joint Symposium on Computer-Aided Control System Design, March 7-9, 1994, Tuscon USA, pp. 315–320
[4] Schneider S.A., Chen V.W., PARDO-CASTELLOTE._ ControlShell: A Real-Time Software Framework. Proc. of the International Conf. on Robotics and Automation, IEEE, San Diego, May 1994
[5] Schwarz J.J.: Langage d'Aide a la Conception d'Applications Multitaches Temps Reel. APII-AFCET, 1992, Vol 26, no 5, Dunod, 355–385
[6] Schwarz J.J., Skubich J.J. , Szwed P. , Maranzana M.: Real Time Multitasking Design with a Graphical Tool . First IEEE Workshop on Real Time Applications. New York May 1993, pp. 33–36
[7] Simon D., Espiau B., Castillo E., Kappellos K.: Computer-Aided Design of a Generic Robot Controller Handling Reactivity and Real-Time Control Issues_. IEEE Trans. on Control System Technology, Vol.1, No 4, pp. 213–229
[8] Szmuc T.: Partial and total relative correctness for analysis of concurrent systems. Applied Mathematics and Computer Science, Vol.1, No.1, pp.27–34
[9] Szmuc T., Szwed P., Schwarz J.J., Skubich J.: Hierarchical Correctness Verification in Multiphase Real-Time Software Design. Proceedings of the 19th IFAC/IFIP Workshop on Real-Time Programming 1994, Reichenau, 22-24 June 1994, Pergamon Press
[10] Szmuc T., Szwed P.: Skrośne tworzenie programów czasu rzeczywistego w systemie VxWorks. Pomiary, Automatyka, Kontrola, nr 9, 1994
[11] Szmuc T., Szwed P.: Towards Automatic Correctness Verification of Real-Time Programs. Applied Mathematics and Computer Science, 1994, No.4, 643–660
[12] Szmuc T., Szwed P.: Graficzna specyfikacja oprogramowania czasu rzeczywistego. Materiały I Ogólnopolskiej Konferencji Oprogramowania Komputerowych Systemów Czasu Rzeczywistego, Wrocław 1994, 54–67
[13] Szwed P.: LACATRE Reference Guide. Opracowanie Instytutu Automatyki, Akademii Górniczo-Hutniczej, Kraków, 1994
[14] Ward P., Mellor S.: Structured Development for Real-Time Systems. Prentice Hall, 1985.

Warunki prenumeraty na 1996 r.

Zamówienia na prenumeratę czasopism wydawanych przez Wydawnictwo SIGMA-NOT można składać w dowolnym terminie. Mogą one obejmować dowolny okres czasu, tzn. dotyczyć dowolnej liczby kolejnych zeszytów każdego czasopisma.

Zamawiający może otrzymywać zaprenumerowany przez siebie tytuł począwszy od następnego miesiąca po dokonaniu wpłaty. Zamówienia na zeszyty sprzed daty otrzymania wpłaty będą realizowane w miarę możliwości – z posiadanych zapasów magazynowych.

Warunkiem przyjęcia i realizacji zamówienia jest otrzymanie z banku potwierdzenia dokonania wpłaty przez prenumeratora. Dokument wpłaty jest równoznaczny ze złożeniem zamówienia.

Wpłat na prenumeratę można dokonywać na ogólnie dostępnych blankietach w Urzędach Pocztowych (przekazy pieniężne) lub Bankach (polecenie przelewu), przekazując środki pod adres:
Wydawnictwo SIGMA-NOT Spółka z o.o.
Zakład Kolportażu
00-716 Warszawa, skr. poczt. 1004
konto:
PBK S.A. III 0/Warszawa nr 370015-1573-2720-3-67

Wpłaty na prenumeratę przyjmują także wszystkie urzędy pocztowe nadawczo-odbiorcze oraz doręczyciele na terenie całego kraju

Na blankiecie wpłaty należy czytelnie podać nazwę zamawianego czasopisma, liczbę zamawianych egzemplarzy, okres prenumeraty oraz własny adres.
Na życzenie prenumeratora, zgłoszone np. telefonicznie, Zakład Kolportażu ul. Bartycka 20, 00-950 Warszawa, (telefony: 40-30-86, 40-35-89 oraz 40-00-21 wew. 249, 293, 299) wysyła specjalne blankiety zamówień wraz z aktualną listą tytułów i cennikiem czasopism.

Odbiorcy zagraniczni mogą otrzymywać czasopisma poprzez prenumeratę dewizową (wpłata dokonywana poza granicami Polski w dewizach, wg cennika dewizowego z cenami podanymi w dolarach amerykańskich) lub poprzez zamówioną w kraju prenumeratę ze zleceniem wysyłki za granicę (zamawiający podaje dokładny adres odbiorcy za granicą, dokonując równocześnie wpłaty w wysokości dwukrotnie wyższej niż cena normalnej prenumeraty krajowej).

Egzemplarze archiwalne (sprzedaż przelewowa lub za zaliczeniem pocztowym) można zamawiać pisemnie, kierując zamówienia pod adresem: Wydawnictwo SIGMA NOT, Spółka z o.o. Zakład Kolportażu, 00-716 Warszawa, ul. Bartycka 20, paw. B, tel. 40-37-31, natomiast za gotówkę można je nabyć w Klubie Prasy Technicznej w Warszawie ul. Mazowieckiej 12, tel. 26-80-17.

Istnieje możliwość zaprenumerowania 1 egz. czasopisma po cenie ulgowej przez indywidualnych członków stowarzyszeń naukowo-technicznych zrzeszonych w FSNT oraz przez uczniów zawodowych i studentów szkół wyższych. Blankiet wpłaty na prenumeratę ulgową musi być opatrzony na wszystkich odcinkach pieczęcią koła SNT lub szkoły.

W przypadku zmiany cen w okresie objętym prenumeratą Wydawnictwo zastrzega sobie prawo do wystąpienia o dopłatę różnicy cen oraz prawo do realizowania prenumeraty tylko w pełni opłaconej.
Cena jednego egzemplarza: normalna 5,00 zł (50 000 zł), ulgowa 3,75 zł (37 500 zł)
Wartość prenumeraty w zł:
Normalna: kwartalna 15,00 zł (150 000 zł), półroczna 30,00 zł (300 000 zł), roczna 60,00 zł (600 000 zł)
Ulgowa: kwartalna 11,25 zł (112 500 zł), półroczna 22,50 zł (225 000 zł), roczna 45,00 zł (450 000 zł)

informacje

Informatyka na wyższych uczelniach dla gospodarki narodowej

Politechnika Gdańska – Wydział Elektroniki Telekomunikacji i Informatyki – Zakład Zastosowań Informatyki organizuje w Gdańsku w dn. 22-24 listopada 1996 r. II Konferencję pn. „Informatyka na wyższych uczelniach dla gospodarki narodowej". Tematem przewodnim będą zagadnienia jakości systemów informatycznych. Uczestnicy konferencji będą zajmować się takimi kwestiami, jak: nowoczesne kształcenie na kierunkach informatycznych, współczesne technologie i trendy rozwojowe w informatyce, a także przykłady zastosowania informatyki w różnych dziedzinach gospodarki.

Organizatorzy konferencji liczą na to, że obok wskazań dla modyfikacji programów dydaktycznych, określenia możliwości i ofert uczelni, wzajemnego poznania się, w jej wyniku będą podjęte wspólne prace badawcze i rozwojowe, wśród których problemy jakości systemów informatycznych zajmą poczesne miejsce.

Referaty będą grupowane w następujących blokach tematycznych:
– Problemy nauczania informatyki, w tym zagadnienia i programy kształcenia na kierunkach informatycznych, umiejętności absolwenta informatyki, kształcenie informatyczne studentów innych specjalności;
– Zagadnienia jakości systemów informatycznych – czynniki i miary jakości, problemy specyfikacji wymagań, metody zapewnienia jakości i uwzględniające je metodologie projektowania systemów;
– Nowe technologie w informatyce polskiej i światowej ze szczególnym uwzględnieniem zagadnień jakościowych;
– Zastosowania informatyki w różnych dziedzinach gospodarki i na uczelniach – Systemy, pakiety, programy narzędziowe itp.

Planowana jest sesja panelowa poświęcona problemom zapewnienia jakości.

We wszystkich sprawach związanych z konferencją należy kontaktować się z jej organizatorami: ul. Narutowicza 11/12, 80-952 Gdańsk, tel./faks: (058) 472727, e-mail: konfzzi@pg.gda.pl, http://www.elka.pg.gda.pl//Konferencje/ZZI

Konferencja nt. systemów rachunkowości

W Kołobrzegu, w dniach 23-25 września 1996 roku Komisja Rady Naukowej Stowarzyszenia Księgowych w Polsce i Oddział Okręgowy Stowarzyszenia Księgowych w Polsce w Szczecinie (AL. Jedności Narodowej 11) organizują konferencję na temat atestacji, normowania systemów informatycznych rachunkowości. Organizatorzy przewidują przedstawienie do dyskusji wstępnych wyników prac Zespołu Roboczego Stowarzyszenia Księgowych w Polsce i Polskiego Towarzystwa Informatycznego, powołanego do opracowania wymagań funkcjonalnych i eksploatacyjnych stawianych systemom informatycznym rachunkowości.

W sprawach organizacyjnych należy kontaktować się z p. dyrektor Ireną Chylewską – tel. (091) 339 832. W sprawach merytorycznych należy kierować korespondencje na:
idziedz@uoo.univ.szczecin.pl

KNSŁ-96

Wyższa Szkoła Oficerska Wojsk Łączności, Wojskowy Instytut Łączności i Wojskowa Akademia Techniczna organizują w Zegrzu w dn. 02-03 października 1996 r. V Konferencję Naukową na temat: „Systemy łączności i informatyki na potrzeby obrony i bezpieczeństwa RP". Na konferencji omawiane będą następujące tematy:
– Teoretyczne podstawy systemów łączności i informatyki;
– Organizacja i eksploatacja systemów łączności i informatyki;
– Problemy bezpieczeństwa systemów łączności i informatyki;
– Projektowanie, produkcja i wdrażanie urządzeń oraz systemów łączności i informatyki.

Obrady odbywać się będą w sesjach plenarnych i sekcjach problemowych. Konferencji towarzyszyć będzie wystawa urządzeń komputerowych, pomiarowych i systemów łączności. Na specjalnej sesji firmy będą prezentować swoje oryginalne rozwiązania.

Informacji na temat konferencji udziela płk dr inż. Antoni Wieczorek, Wyższa Szkoła Oficerska Wojsk Łączności, 05-131 Zegrze 3, tel. (022) 6883314, faks – (022) 6883400.

Visual dBase 5.5 Professional

Borland przygotował kolejną wersję pakietu Visual dBase – jedynej obiektowej bazy danych standardu Xbase drugiej generacji. Po wersji podstawowej: klient serwer z kompilatorem i narzędziami umożliwiającymi dostęp do serwerów SQL, obecnie oferowana jest wersja Professional. Oprócz wsparcia dla Windows 3.1 i Windows 95 Visual dBase wyposażony jest w narzędzie typu Two-Way, a w wersji Professional dodatkowo znajduje się kompilator i pakiet Intranet Tools, umożliwiający dołączenie baz danych do stron WWW. Dostęp można uzyskać z dowolnej platformy sprzętowej, korzystając ze standardowej przeglądarki WWW. Intranet Tools pozwala na korzystanie z danych w dowolnym formacie obsługiwanych przez Borland Database Engine włączając dBase, Paradox, ODBC, Oracle, Microsoft SQL Server, Informix i Borland InterBase. Visual dBase Intranet Tools składają się z: Table Publish Web Expert, Table Search Web Expert i Delta Point Web Tools for Visual dBase. Umożliwiają one m.in. automatyczne przekształcanie tabel w strony HTML, tworzenie zapytań i wyszukiwanie danych. Interfejs CGI i narzędzia do przygotowania stron WWW umożliwiają tworzenie serwerów WWW zorientowanych na zarządzanie bazami danych. Pakiet zawiera kod źródłowy bibliotek, opisy klas, metod i funkcji oraz wiele przykładów.

Ulepszone drukarki firmy Xerox

Seria biurowych drukarek laserowych Xerox XPrint przekształciła się w serię XPrint Plus, w której podwyższenie jakości wydruku bez podnoszenia ceny stało się możliwe dzięki nowym koncepcjom oprogramowania Xerox Intelligent Colour 2.0. Zastosowana w nim technologia Photoreal powoduje, że urządzenie automatycznie rozpoznaje typ zadania (obraz, wykres, tekst itp.) i dobierając odpowiednie procedury obróbki zapewnia m.in. realistyczne odtwarzanie fotografii (wyeliminowany efekt mory), ostrzejszy kontur i żywszy kolor wykresu lub tekstu. Seria Plus jest zgodna ze standardem ColorSync 2.0 oraz Windows Integrated Color Management. Seria zawiera modele 4915 (najtańsza kolorowa drukarka laserowa w Europie), 4920 i 4925. Oferują one rozdzielczość 600x600 dpi (4915 ma opcję 12000x300 dpi osiąganą na drodze modulacji światła laserowego), drukują z szybkością 3 str./min przy pełnej barwie i 12 str./min przy wydruku monochromatycznym. Drukarki XPrint Plus są przystosowane do pracy grupowej.

Ofensywa IBM na rynku PC

IBM sprzedał w ub.r. w Polsce nie 8 tys. PC-ów, jak podano w Raporcie Specjalnym Computerworld TOP 200, lecz 17 tys. – oświadczył Paweł Maksyś, dyrektor d.s sprzedaży komputerów osobistych w IBM Central & Eastern Europe, na konferencji prasowej IBM, poświęconej prezentacji najnowszych modeli komputerów osobistych. Stawia to IBM na trzecim miejscu za Optimusem i JTT, a przed największym rywalem – firmą Compaq.

opracowała: Krystyna Karwicka

C570 – przemysłowy PC do wbudowania

Firma PEP Modular Computers wprowadziła do sprzedaży nowy przemysłowy komputer klasy PC typu C570, który został zbudowany w formie panelu o wymiarach 330x266 mm. Jest on przeznaczony do pracy w najcięższych warunkach przemysłowych. Tak dobre warunki eksploatacyjne osiągnięto dzięki oparciu konstrukcji panelu o 7 mm płytę przednią, w którą wbudowany jest wyświetlacz i foliowa klawiatura. Zaletą C570 jest nie tylko mocna konstrukcja, ale i pełna zgodność z PC. Komputery wyposażone są w procesor 486DX4 lub Pentium, pamięć operacyjną od 4 do 128 MB, dysk minimum 420 MB z zawieszeniem antywibracyjnym. C570 mogą być wyposażone w wyświetlacz LCD mono lub TFT kolor. Komputer C570 zaprojektowany został z myślą o zastosowaniach typu panel operatorski, do wbudowania w ścianę, pulpit czy też maszynę, czyli wszędzie tam, gdzie czas przeżycia standardowego PC sięgałby kilku chwil.

*

Firma Novell wprowadziła na rynek oprogramowanie LAN WorkPlace Pro for Windows 95/NT – nową wersję zintegrowanego zestawu aplikacji współpracujących z protokołem TCP/IP (Transmission Control Protocol/Internet Protocol), pozwalającego użytkownikom uzyskiwać dostęp do danych i aplikacji, korzystać z nich wspólnie i kopiować je przez sieci Intranet oraz Internet. Nowa wersja stanowi rozszerzenie serii produktów LAN WorkPlace Novella, w skład której wchodzą już produkty dla systemów DOS i Windows 3.x. Korzysta z nich ponad 3,5 mln użytkowników.

*

Apple Computer podpisał umowę licencyjną z IBM, która daje IBM–owi prawo do sublicencjonowania systemu Mac OS każdemu producentowi systemów IBM, opartych na mikroprocesorach PowerPC. Pozwoli to na rozwijanie i dostarczanie systemu operacyjnego Macintosh OS wraz z szerokim asortymentem systemów opartych na obecnej architekturze Power Macintosh, a w niedalekiej już przyszłości wraz z maszynami wyprodukowanymi w oparciu o wspólną specyfikację o nazwie PowerPC Platform (CHRP).

*

Firma KYE Systems Corporation – producent elementów sieciowych, znanych pod nazwą GeniusLan – wprowadziła na polski rynek za pośrednictwem dystrybutora – JTT – nową rodzinę 16-bitowych kart sieciowych standardu Ethernet. Oznaczona jest ona symbolem GE200III i składa się z trzech typów kart różniących się między sobą typami gniazd. Istnieje możliwość rozszerzenia każdej z tych kart o złącze AUI. Nowo wprowadzone karty mają zastąpić dotychczas produkowane modele: GE2000II-CX, GE2000II-TP, GE2000II-2.

*

Firma Intergraph Software Solutions zawarła z Microsoftem porozumienie o pogłębieniu współpracy obu firm w zakresie produktów integrujących środowiska Windows i Unixa, zwłaszcza włączania systemu plików NFS (Network File System) do Windows NT. W III kwartale br. wchodzą do sprzedaży nowe wersje dwu programów pracujących w środowisku Windows NT: DiskAccess 3.0, który pozwala użytkownikom Windows NT i Windows 95 korzystać z plików danych składowanych na serwerach unixowych i DiskShare 3.0, który powoduje, że systemy baz danych osadzonych w Windows NT są dostępne dla użytkowników Unixa. Oba programy obsługują platformy Intel, MIPS, Alpha-AXP i PowerPC. Wersje beta obu programów są osiągalne w sieci WWW pod adresem: http://www.intergraph.com/nfs

*

Compaq wprowadził na rynki Europy, Bliskiego Wschodu i Afryki serię wyrobów sieciowych Netelligent, przeznaczonych dla biur filialnych i terenowych, dla zespołów wydziałów oraz małych i średnich firm. Seria zawiera ponad 100 osobnych, standardowych produktów sieciowych, zapewniających łączność w przedsiębiorstwie, z których większość Compaq zakupił w ub.r. w firmach Thomas-Conrad i Net-Worth. Są też cztery nowe produkty Netelligent: przełącznik 5506 100Base-T, sterowniki sieciowe 10Base -T i 10/100Base-TX, sterownik sieciowy Token Ring PCI.

*

Intel zaprezentował nową technologię komunikacji wideo, przeznaczoną na rynek domowych komputerów PC. Opiera się ona na linii wyrobów do prowadzenia wideokonferencji, ProShare. Dzięki wykorzystaniu w Intel Video Phone technologii ProShare, klienci mogą za pomocą komputera osobistego z procesorem Pentium o częstotliwości zegara 133 MHz i wyższej rozszerzyć zwykłą rozmowę telefoniczną o obraz rozmówcy. Intel Video Phone jest zgodny ze standardami telekomunikacyjnymi, w szczególności H.324.

*

Firma Autodesk skierowała do sprzedaży dwa dyski optyczne – 3D Props Residential i 3D Props Commercial – zawierające ponad 300 bardzo szczegółowych i gotowych do wykorzystania, trójwymiarowych obiektów, przydatnych do tworzenia modeli i animacji. Ich rozpowszechnianie nie podlega opłacie. Są to pierwsze pozycje z serii Multimedia Clip Libraries Autodesk Data Publishing, dzięki której Autodesk chce zdobyć pozycję czołowego wydawcy materiałów cyfrowych dla projektantów, kreślarzy, producentów i autorów pozycji multimedialnych.

*

ICL prowadzi intensywne prace nad rozwojem systemu operacyjnego NX, zgodnego ze standardem Unix. Opracowywany jest system NXV7 Mplus, przeznaczony dla serwerów opartych na procesorach firmy SPARC oraz system UnixWare, przeznaczony dla serwerów opartych na procesorach firmy Intel.

*

Firma American Power Conversion wprowadziła do sprzedaży zewnętrzne urządzenie SmartSlot Chassis, które służy do instalowania kart rozszerzających o nazwie SmartSlot, produkowanych przez APC, jeżeli UPS nie jest wyposażony w gniazdo umożliwiające bezpośrednie umieszczenie w nim karty. Smart Slot jest nową technologią wprowadzoną do UPS-ów APC, która pozwala na instalowanie specjalnych kart rozszerzających, umożliwiających np. zarządzanie UPS-em przez protokół SNMP lub monitorowanie parametrów środowiska zewnętrznego.

*

Microsoft SQL Server, działający na serwerze Intergraph InterServe MP – 610, oraz Microsoft Internet Information Server, działający na serwerze Intergraph Web 300 (oba serwery z mikroprocesorem Intel Pentium Pro), osiągnęły w pierwszym, opartym na Internecie, teście C Transaction Processing Council (TPC-C) bardzo dobry stosunek ceny do wydajności: 124 USD/tpmC (transakcji na minutę). Zanotowały też rekordową przepustowość w kategorii systemów jednoprocesorowych (1 675 tpmC). Wprowadzony na rynek w kwietniu Microsoft SQL Server 6.5 jest najszybszą bazą danych do sieciowego systemu operacyjnego Windows NT Server. W połączeniu z Microsoft Internet Information Server zapewnia duże możliwości tworzenia publikacji do sieci WWW.

*

Western Digital Corporation wyprodukowała nowy dysk – AC33100 (trzy talerzowy, 3.1 GB) z rodziny napędów Caviar EIDE. Charakteryzuje się on średnim czasem odczytu, mniejszym niż 12 ms i czasem oczekiwania wrzeciona 5.76 ms. Zapewnia to szybki dostęp mechaniczny do danych pobieranych bezpośrednio i za pośrednictwem pamięci cache. Napęd może pracować zarówno w Mode 4 PIO (programowane wejście/wyjście), jak i w Mode 2 DMA (bezpośredni dostęp do pamięci). Szybkość przesyłania danych dochodzi do 16.6 Mb/s przy prędkości wirowania 5200 obr/min.

opracowała: Krystyna Karwicka

Kanon czy rynek? Co ma być wyznacznikiem kształcenia informatyków?

Andrzej Pająk

Powyższe pytania w odniesieniu do dziedzin podstawowych, takich jak nauki przyrodnicze czy matematyka, mogłyby zostać potraktowane jako obraźliwe. Dziedziny naukowe zajmujące się odkrywaniem i poznawaniem praw przyrodniczych lub obiektów abstrakcyjnych, ze swoimi teoriami, metodami badawczymi, długimi wykazami hipotez i problemów oczekujących na rozwiązanie, są mało wrażliwe na bezpośrednie działanie niewidzialnej ręki rynku. Ale na pewno są wrażliwe na poziom finansowania, co ma z rynkiem bezpośredni związek. Dziedziny techniczne, w tym również szeroko rozumiana informatyka, są postrzegane przede wszystkim przez rynek dostępnych produktów danej branży i związany z tym rynek pracy. Nie znaczy to bynajmniej, że dla tych dziedzin głównym celem jest tylko zadowolenie konsumenta, gotowego kupić następną, ulepszoną wersję produktu, by poczuć się jeszcze lepiej. Dziedziny techniczne, w moim rozumieniu, to dziedziny dwuwarstwowe: w warstwie głębszej dokonuje się postęp, powstają nowe teorie, metody i technologie; w warstwie zewnętrznej wytwarza się produkty i robi interesy.

Informatyka, jak i inne dziedziny techniczne, ma swoją bazę naukową i metodyczną wedle której formułuje się, między innymi, zasadniczy kanon kształcenia w profesji. Ludzie odpowiedzialni za opracowanie programów kształcenia informatyków w różnych ośrodkach akademickich w Polsce (i gdzie indziej) zapewne właściwie kojarzą lata 1968, 1978, 1991. W tych latach ukazywały się w USA kolejne zalecenia programowe dla uniwersytetów zajmujących się kształceniem informatyków. Ostatnie w tej serii opracowań *(ACM/IEEE-CS Curricula 1991)* poprzedzone zostało raportem wybitnych specjalistów pod przewodnictwem Petera Denninga, w którym przedstawiono nową, w pewnym sensie wyliczeniową definicję informatyki (to znaczy jej warstwy głębszej). Raport nosił tytuł *„Computing as a Discipline"* i oprócz konkretyzacji poszczególnych obszarów tematycznych systematyzował również strukturę naukową dziedziny. *Curricula 91* opracowano traktując raport jako intelektualną podstawę organizacji programu kształcenia i określenia obligatoryjnego kanonu wiedzy.

Co w sprawie kształcenia mówi rynek pracy? Nie znam żadnych miarodajnych i udokumentowanych ocen absolwentów z punktu widzenia oczekiwań firm komputerowych, ale słyszy się często zarzut o braku (bezpośrednio użytecznych dla firmy) umiejętności posługiwania się pewnymi narzędziami, administrowania systemami, pracy projektowej w zespole, kompetencji w zakresie integracji systemów/sieci, itp. Śmiem twierdzić, że polski rynek pracy informatyków oczekuje od absolwentów więcej po stronie gotowych umiejętności niż przygotowania podstawowego i otwartości intelektualnej. Nie jest to zarzut pod adresem firm, lecz konstatacja stosunkowo łatwego do wytłumaczenia faktu. Firma mała i na dorobku, typowa na rynku polskim, nie chce albo nie może sobie pozwolić na zatrudnienie pracownika, który nie potrafi od razu wykonać pracy przynoszącej zysk firmie. Inwestowanie w pracowników jest dobrą lokatą, ale wymaga kapitału. Firma duża, zaangażowana również w rozwój nowych metod i technologii, będzie bardziej zainteresowana zatrudnieniem takiego pracownika, który mimo braku pewnych umiejętności w tej chwili może być innowacyjny w niedalekiej przyszłości.

Jak zatem powinny zachować się uczelnie? Kompromisy bywają niekiedy kiepskim rozwiązaniem, ale w tym przypadku innego nie ma. Byłoby błędem, gdyby uniwersytety i uczelnie techniczne wyniośle odrzuciły sygnały dochodzące od pracodawców i studentów, traktując swoją misję edukacyjną jako obronę depozytu kanonu. Dla wykładowcy może być niemiła konstatacja, że studenci chętniej wybierają przedmiot poświęcony konkretnemu pakietowi oprogramowania niż bardziej głęboki przedmiot z zakresu języków formalnych i podstaw teoretycznych technik kompilacji. Z drugiej strony, jeśli uczelnie traktują poważnie również swoje zadania badawcze, to nie mogą pominąć kształcenia podstawowego w dziedzinie.

Studia dwustopniowe (w uczelniach technicznych inżynierskie i magisterskie) stanowią formę organizacyjną realizacji tego kompromisu. Sama forma oczywiście nie wystarcza. W tym miejscu powinienem zakończyć notatkę, ponieważ doszliśmy do kwestii pieniędzy, odpływu wykwalifikowanych kadr, luki pokoleniowej i pozostałych czterech plag szkolnictwa wyższego. Na ten temat napisano i powiedziano bezskutecznie już tyle, że dalsze przywoływanie argumentów na rzecz samej edukacji narodowej prawdopodobnie nikogo z decydentów nie poruszy. Zamiast tego pozwolę sobie zwrócić się z apelem do firm komputerowych: jesteśmy dla siebie partnerami – powinno nam wzajemnie na sobie zależeć.

Wiesbaden '96

W Wiesbaden w dniach 9-11.06.1996 r. odbyła się międzynarodowa konferencja Software AG. Temat sympozjum ujęto w sformułowanie „Technika informacyjna z ludzka twarzą". Przedstawiciele administracji publicznej, mediów i producenci oprogramowania dyskutowali na temat przyszłości technik informacyjnych i ich wpływu na społeczeństwo. W konferencji brali udział m. in. Martin Bangemann, Lothar Klemm, Peter Keen. Z Polski zaproszono: Leszka Balcerowicza, Wacława Iszkowskiego oraz Marcina Łakomego. **(ap)**

informatyka

Miesięcznik
ISSN 0542-9951
INDEKS 36124

Uwaga
ukazuje się już elektroniczna wersja naszego miesięcznika. Oto nasz adres w Internecie:
http://www.pol.pl/informatyka/

nr 8 1996
Sierpień
Rok wydania XXXII

REDAGUJE ZESPÓŁ:
dr Lesław **WAWRZONEK**
(redaktor naczelny)
mgr Krystyna
KARWICKA-RYCHLEWICZ
(z-ca redaktora naczelnego)
mgr Jerzy **SZYLLER**
(redaktor)
mgr Zdzisław **ŻURAKOWSKI**
(redaktor działu)
Agnieszka **PAPIS**
(sekretarz redakcji)
Alina **KLEPACZ**
(administracja redakcji)

KOLEGIUM REDAKCYJNE:
prof. dr hab. Leonard BOLC,
mgr inż. Piotr FUGLEWICZ,
prof. dr hab. Jan GOLIŃSKI,
dr inż. Zenon KULPA,
prof. dr inż. Jan MULAWKA,
prof. dr hab. Wojciech OLEJNICZAK,
mgr inż. Jan RYŻKO,
dr Witold STANISZKIS,
dr inż. Jacek STOCHLAK,
prof. dr hab. Maciej STOLARSKI,
prof. dr hab. Zdzisław SZYJEWSKI,
prof. dr hab. inż.
Ryszard TADEUSIEWICZ,
prof. dr hab. Jan WĘGLARZ

PRZEWODNICZĄCY
RADY PROGRAMOWEJ
prof. dr hab.
Juliusz Lech KULIKOWSKI

WYDAWCA:
Wydawnictwo Czasopism i Książek Technicznych SIGMA NOT
Spółka z o.o.
ul. Ratuszowa 11
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
00-950 Warszawa,
ul. Ratuszowa 11, p. 628 i 644
skrytka pocztowa 1004
tel., fax: 619-11-61
tel.: 619-22-41 w. 159
e-mail:
informat@pol.pl

Materiałów nie zamówionych redakcja nie zwraca

Autorzy artykułów proszeni są o przysyłanie tekstów na dyskietkach $3^1/_2$" – w czystych kodach ASCII (Latin II, Mazovia) lub edytorach: Word 2, Word 6 – oraz odbitki na papierze. Teksty nie mogą przekraczać 20 tys. znaków.

Redakcja nie ingeruje w treść i formę ogłoszeń i innych materiałów reklamowych, w związku z tym nie ponosi za nie odpowiedzialności.

Ogłoszenia przyjmują:

- **Redakcja**
- **Dział Reklamy i Marketingu**
  00-950 Warszawa,
  ul. Mazowiecka 12
  tel.: 27-43-66,
  fax: 26-80-16

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o. z. 290/96

W numerze :

publikacje

OB! OK! - obiektowy system zarządzania bazą danych 1
Maciej Piasecki

Reprezentacja czasu w modelu konceptualnym SI 5
Krzysztof Kania, Jerzy Gołuchowski

Metody szyfrowania informacji 8
Naceur Frih

Programowanie obiektowe w CA-Clipper 12
Marek Miłosz, Bogdan Wit

Hierarchiczne tworzenie aplikacji czasu rzeczywistego 22
Tomasz Szmuc, Piotr Szwed

felietony

Kanon czy rynek? Co ma być wyznacznikiem kształcenia informatyków? 31
Andrzej Pająk

prezentacje

***GANDALF* stawia na rozwój sieci dostępowych i ISDN** 17
Stanisław Michalak

informacje

29-30

W najbliższych numerach:

- **wywiad z Markiem Gondzio, prezesem Stowarzyszenia PRO**
- **Krzysztof Kania i Jerzy Gołuchowski o systemach temporalnych baz danych**
- **Witold Abramowicz, Grzegorz Bartoszewicz i Leszek A. Maciaszek o rywalizacji i koegzystencji relacyjnych i obiektowych baz danych**

JUNISOFTEX

20 LAT DOŚWIADCZEŃ WDROŻENIOWYCH

Perfect-Ekspert

Sieciowy i zintegrowany system informatyczny pracujący na platformach PC i AS/400, głównie dla dużych firm produkcyjnych i handlowych. Przystosowany do elektronicznej wymiany dokumentów (EDI-zamówienia, faktury, WZ itp.). Wielowalutowości oraz rachunkowości zarządczej. Wyposażony w system ochrony danych generowanych wewnątrz i na zewnątrz przedsiębiorstwa.

Oferujemy kompleksowe usługi w zakresie:
- doradztwa organizacyjnego, merytorycznego i technicznego
- organizacji, projektowania i wykonawstwa innych systemów informatycznych
- wdrażania, konserwacji i modernizacji systemów informatycznych
- projektowania, wykonawstwa dowolnego typu sieci komputerowych, z uwzględnieniem już istniejących rozwiązań
- dostawę sprzętu komputerowego i sieciowego
- szkoleń użytkowników w zakresie operatora i administratora systemu AS/400, jak również obsługi pakietów programowych PC Support 400 lub Client Acces /400 przeznaczonych do współpracy z PC

Elektrownia RYBNIK
Zakłady Rowerowe "ROMET"
Krywałd "ERG" S.A.
Huta Łaziska
Huta Bankowa
Huta 1-go Maja
Huta Szkła Kunice
Zakłady Metalurgiczne Trzebinia
Walcownia Metali "Dziedzice"
Zakłady Górniczo - Hutnicze "BOLESŁAW"
Zakłady Gumowe Górnictwa
Fabryka Osi Napędowych
MOSTOSTAL
Elektromontaż 3
POLIFARB Cieszyn S.A.
POLAM Kostuchna
PKM Katowice
WEDEL S.A.
Instytut Informatyki Teoretycznej i Stosowanej
Centrum Naukowo Techniczne Kolejnictwa
Instytut Spawalnictwa

PRZEDSIĘBIORSTWO PRODUKCYJNO-USŁUGOWE INNOWACJI TECHNICZNYCH JUNISOFTEX
BIURO TECHNICZNO-HANDLOWE, GLIWICE, ul. Kościuszki 1c, box 507, tel./fax: (0-32) 1306931, 1306932, 1306933, 1306934, telex 0316333

Polska OnLine

INTERNET

http://www.pol.pl

JESTEŚ AKTYWNY!
BĄDŹ INTERAKTYWNY!

Polska OnLine Sp. z o.o. 01-756 Warszawa, ul.Przasnyska 6,
tel. 639 85 45, fax 663 52 81, e-mail: info@pol.pl